

2018-05-23

New Approaches for Secure Distance- Bounding

Ahmadi Fatlaki, Ahmad

Ahmadi Fatlaki, A. (2018). New Approaches for Secure Distance- Bounding (Doctoral thesis, University of Calgary, Calgary, Canada). Retrieved from <https://prism.ucalgary.ca>. doi:10.11575/PRISM/31947
<http://hdl.handle.net/1880/106680>

Downloaded from PRISM Repository, University of Calgary

UNIVERSITY OF CALGARY

New Approaches for Secure Distance- Bounding

by

Ahmad Ahmadi Fatlaki

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

May, 2018

© Ahmad Ahmadi Fatlaki 2018

Abstract

In this thesis we design and implement three aspects of secure distance-bounding (DB) schemes as a type of authentication scheme that considers distance as an extra verification parameter. By adding this new parameter to authentication schemes, we can prevent certain attacks that are related to distance, such as relay attack. In fact, the attacking scenarios can be much more complex than the simple relay attack, in addition to the classic authentication scheme attacks. In this thesis we consider the most advanced distance-bounding attack scenarios in a variety of authentication models. We consider three authentication models in order to add the distance as an extra authentication factor: public-key and anonymous DB are the main fields of this thesis that consider strong adversary with access to directional antenna, and we consider One-Shot DB as a one-message authentication scheme. Each of these fields make a chapter of this thesis.

Public-Key Distance-Bounding. In a public-key DB scheme, a prover who owns a key pair and is located within a distance bound to a verifier, who has access to the public-key of the prover, tries to convince the verifier that it is authentic and located within the distance bound. We provide a formal model and two protocols with security proofs.

Anonymous Distance-Bounding. In an anonymous DB scheme, a prover who owns a registration certificate and is located within a distance bound to a verifier, who only has access to the public parameters of the system, tries to convince the verifier that it is authentic and located within the distance bound without revealing its identity. We provide a formal model and two secure protocols.

One-Shot Distance-Bounding. In an one-shot DB scheme, a prover who owns a secret key and is located within a distance bound to a verifier, who has access to the corresponding key of the prover, tries to convince the verifier that it is authentic and located within the distance bound without receiving any message from the verifier. We provide a formal model and a secure protocol.

Acknowledgements

I want to express how lucky I've been in life and specifically in the last era of my life that finishes by this thesis. I've been gifted by some intelligence that helped me to generate ideas, great environment that continuously exposed me to infinite knowledge, and learning abilities that helped me to absorb some of them. Most importantly, I've been gifted by the feeling of appreciation to those who gave these gifts to me.

First, I want to express my deepest thanks to God who provided existence to me with all these shiny gifts. I'm thankful for the good and the bad of life that each taught me a lesson.

I want to thank my parents for always being there for me and believing in me. My patient dad, Nemat Ahmadi, who taught me how to feel the power of free will. My loving mom, Goldasteh Ebrahimpour, who taught me how to enjoy caring for people, by always putting my happiness above hers. They provided a peaceful environment for my soul to grow.

I want to thank my PhD supervisor, Reyhaneh Safavi-Naini, who always cared for me by teaching me discipline, hard working, taking responsibility, and having respect for my own time and others'. It was a great opportunity for me to learn directly from her.

I don't know if I deserve all these gifts, but I'm just so happy for having them :)

Table of Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Tables	vii
List of Figures	viii
List of Symbols	xi
1 Introduction	1
1.1 Public-Key Distance-Bounding	6
1.2 Anonymous Distance-Bounding	7
1.3 One-Shot Distance-Bounding	8
1.4 Thesis Organization	11
2 Background and Notations	12
2.1 Encryption	13
2.2 Signature	15
2.3 Authentication Protocol	16
2.4 Identification	19
2.5 Proof Systems	23
2.6 Commitment	26
2.7 Theorems	29
2.8 Concluding Remarks	30
3 Literature Review	31
3.1 Distance-Bounding as Challenge-Response Entity Authentication	32
3.1.1 Symmetric Key Distance-Bounding	35
3.1.2 Public-Key Distance-Bounding	38
3.1.3 Anonymous Distance-Bounding	45
3.1.4 Concluding Remarks	49
3.2 Time-based Entity Authentication	52
3.2.1 Bellare-Rogaway Model [BR93]	52
3.2.2 Schwenk [Sch14]	54
3.2.3 Barbosa-Farshim [BF09]	57
3.2.4 Concluding Remarks	60
3.3 Other Literature Reviews	61
4 Public-Key Distance-Bounding	62
4.1 Directional Attacks on Public-Key DB Protocols	65
4.2 Model	72
4.3 DBID Construction: POXY	87
4.3.1 $(msk, gpk) \leftarrow \text{Init}(1^\lambda)$	87
4.3.2 $(sk, pk) \leftarrow \text{KeyGen}(msk, gpk)$	88
4.3.3 $\text{accept/reject} \leftarrow \Pi\{P(sk, pk) \leftrightarrow V(pk)\}$	88
4.3.4 $(msk', gpk') \leftarrow \text{Revoke}(msk, gpk, i)$	90
4.3.5 Security Analysis	90
4.4 DBID Construction: ProProx [Vau14]	98

4.4.1	$(msk, gpk) \leftarrow \text{Init}(1^\lambda)$	99
4.4.2	$(sk, pk) \leftarrow \text{KeyGen}(msk, gpk)$	99
4.4.3	$\text{accept/reject} \leftarrow \Pi\{P(sk, pk) \leftrightarrow V(pk)\}$	99
4.4.4	$(msk', gpk') \leftarrow \text{Revoke}(msk, gpk, i)$	99
4.4.5	Security Analysis	100
4.5	Related Works	106
4.6	Concluding Remarks	109
5	Anonymous Distance-Bounding	110
5.1	New Attacks	113
5.1.1	Directional TF Attack on Anonymous DB	114
5.1.2	Collusion TF on Anonymous DB	119
5.2	Model	121
5.3	AnonDB Construction: $\text{dbid2an}^{\text{GM}}$	134
5.3.1	$(msk, gpk) \leftarrow \text{Init}(1^\lambda)$	136
5.3.2	$(s, msk', gpk') \leftarrow \text{CertGen}(msk, gpk)$	136
5.3.3	$\text{accept/reject} \leftarrow \text{CertVer}(s, gpk)$	137
5.3.4	$\text{accept/reject} \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$	137
5.3.5	$(l) \leftarrow \text{Open}(msk, \text{transcript})$	138
5.3.6	$(msk', gpk') \leftarrow \text{Revoke}(msk, gpk, l)$	139
5.3.7	Security Analysis	139
5.4	AnonDB Construction: $\text{dbid2an}^{\text{P}}$	148
5.4.1	$(msk, gpk) \leftarrow \text{Init}(1^\lambda)$	149
5.4.2	$(s, msk', gpk') \leftarrow \text{CertGen}(msk, gpk)$	149
5.4.3	$\text{accept/reject} \leftarrow \text{CertVer}(s, gpk)$	150
5.4.4	$\text{accept/reject} \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$	150
5.4.5	Security Analysis	151
5.5	Related Works	158
5.6	Concluding Remarks	159
6	One-Shot Distance-Bounding	160
6.1	Model	161
6.1.1	Protocol	162
6.1.2	Participants and Process Oracles	164
6.1.3	Adversary	165
6.1.4	Running the protocol in presence of the adversary	166
6.2	One-Shot DB Attacks	168
6.3	Beacon-based Secure One-Shot DB Scheme	172
6.3.1	Global Beacon	173
6.3.2	Adversary	175
6.4	Beacon-based One-Shot DB Construction: BShot	176
6.4.1	BShot Accuracy	179
6.4.2	BShot Practicality	180
6.4.3	BShot Security Analysis	183
6.5	Concluding Remarks	187
7	Concluding Remarks	188
	Bibliography	190

A	Extra Literature Review	202
A.1	Symmetric Distance-Bounding	202
	A.1.1 Formalizing Security in Distance-Bounding	210
A.2	Implementation of Distance-Bounding	211
A.3	Privacy in Distance-Bounding	217

List of Tables

3.1	Security of different DB protocols. The first column shows the name of protocol and its type S ymmetric-key/ P ublic-key/ A nonymous. The fifth column shows if the protocol is noise resistant or not. The last column shows who the prover is anonymous to: E avesdropper, MiM , or V erifier. Two colored cells with same color are referring to a single paper. The symbol \Downarrow denotes this thesis.	51
A.1	Vulnerable DB protocols against PRF Programming Techniques [BMV12].	206

List of Figures and Illustrations

1.1	Legitimate Access	1
1.2	Fraudulent Access	1
1.3	Generic DB protocol. In shared-key setting, $y = x$	2
1.4	Mobile participants in DB	9
3.1	Distance Fraud Attack	33
3.2	Mafia Fraud Attack	34
3.3	Terrorist Fraud Attack	34
3.4	Distance Hijacking Attack	35
3.5	Impersonation Attack	35
3.6	General sketch of symmetric key distance-bounding protocols. x is the shared key with length λ . In initialization phase, the parties exchange randomness, that allows them to generate a table. In fast phase, the verifier asks about half values of the table and measures response time. In verification phase, the verifier checks validity of responses in respect to the table.	36
3.7	Swiss-Knife distance-bounding protocol [KAK ⁺ 08]. x is the shared key with length λ , ID is identity of prover and $\{ID\}$ is identity list of all provers. $f_x()$ is a pseudo random function.	37
3.8	Brands-Chaum Protocol [BC94]. x is prover's private key with length λ and y is prover's public key. Prover commits to a random table and then runs the fast phase. In verification phase, prover opens the committed table.	39
3.9	DBPK-log Protocol. x is prover's private key with length λ and y is prover's public key. t and λ are security parameters.	40
3.10	HPO Protocol. x_j is prover's private key with length λ and X_j is prover's public key. y is verifier's private key and Y is verifier's public key. $DB = \{X_j\}$ is public-key list of all provers. G is a generator point on elliptic curve and $xcoord(.)$ returns X coordinate of a point, which is statistically indistinguishable from the uniform distribution.	42
3.11	VSSDB public-key DB protocol. (sk_p, x) is prover's private key where x has random distribution with length λ and pk_p is prover's public key. sk_v is verifier's private key and pk_v is verifier's public key. (Commit, COpen) is a commitment scheme. (Enc, Dec) is a secure public key encryption scheme. (Sign, SVerify) is a signature scheme. (Prove, PVerify) is a proof-of-knowledge scheme. H is a secure hash function with pseudo-random output. $v_j = H^j(x)$ and $com_j = \text{Commit}(x_j, v_j)$ for $j = 1 \dots \lambda$	43
3.12	ProProx public-key DB protocol. sk is prover's private with length λ and pk is prover's public key. $Com(.,.)$ is Goldwasser-Micali encryption. τ is the minimum threshold ratio of noiseless fast rounds. ZKP is an interactive zero-knowledge proof. The number of fast rounds is $n\lambda$. In each fast round, the verifier sends one-bit challenge, and receives the corresponding response.	44

3.13	GOR anonymous DB protocol. λ is security parameter. sk_j is prover's private and PK_j is prover's public key. sk_v is verifier's private and PK_v is verifier's public key. $(HEnc, HDec)$ is a homomorphic encryption scheme. $(Prove_{NIZK}, Verify_{NIZK})$ is a non-interactive zero-knowledge proof system. PRF is a pseudorandom functions. G is a generator point on elliptic curve and $xcoord(.)$ returns X coordinate of a point. $PK_v = sk_v G$; $PK_j = \prod_{i=1, i \neq j}^k sk_i G$; $Q = \prod_{i=1}^k sk_i G$	47
3.14	SPADE anonymous DB protocol. λ is security parameter. sk_p is prover's private key and gpk is group public key. sk_v is verifier's private and pk_v is verifier's public key. $(GSign_{sk}, GVerify_{gpk})$ is a group signature scheme. (Enc_{pk}, Dec_{sk}) is a secure public-key encryption scheme. $PRF : \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \rightarrow \mathbb{Z}_2^\lambda$ and $PRF^* : \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \rightarrow \mathbb{Z}_2^\lambda$ are pseudo-random functions. λ is the security parameter. N_p and (N_v, m) are nonce values of prover and verifier, and (c_i, r_i) is a challenge and response round.	48
3.15	TREAD public-key/anonymous DB protocol. λ is security parameter. sk_p is prover's private key and gpk is group public key. sk_v is verifier's private and pk_v is verifier's public key. $(GSign_{sk}, GVerify_{gpk})$ is a group signature scheme. (Enc_{pk_v}, Dec_{sk_v}) is a secure encryption scheme. id_{group} is name of the group that the prover belongs to for anonymous authentication.	49
4.1	Directional TF	65
4.2	Step (i). DBPoK-log ⁺ Bit Commitment. r_0 and r_1 form the response table.	68
4.3	Step (ii). DBPoK-log ⁺ Fast Challenge/Response	69
4.4	Step (iii). DBPoK-log ⁺ Commitment Opening	70
4.5	89
5.1	Collusion DF	111
5.2	Collusion MF	112
5.3	Collusion TF	112
5.4	Collusion TF Attack Type 1	120
5.5	Collusion TF Attack Type 2	120
5.6	Π protocol in dbid2an ^{GM} scheme. $C = Com_{H_e}(x, v)$ using Equation 5.3.1.	138
5.7	Π protocol in dbid2an ^P scheme for the l^{th} user. $Commit^P(x, \Delta)$ is the Pedersen commitment function as $Commit^P(x, \Delta) = g^x h^\Delta \bmod p$ (Algorithm 2.6.1). Note that we are using a non-interactive protocol CLSig.SPK, which allows us to break down the protocol into two pieces $CLSig.SPK = (SPK, Verify)$. Note that the value of C is embedded inside π , and we use the notation $C \leftarrow \pi$ to indicate the extraction of C from π . An instance of DBID ^P . Π sub-protocol is shown in Figure 4.5.	151
6.1	Arrival time of a timestamp at different entities. t_1, t_2, t_3 are three consecutive beacon times. GB's timestamp is received at P_2 with δ second delay compared to P_1 . P_1 receives timestamp t_1 , and forms the message (t_1, m) and sends to P_2 . Distance $(d(loc_{P_1}, loc_{P_2}))$ calculated by P_2 is $(t_2 - t_1)/\mathcal{C}$	174
6.2	global beacon broadcasts unpredictable timestamps to users	176
6.3	BShot: One-shot distance bounding protocol. The query server can be replaced by a local memory at each participant. \mathcal{C} is the speed of light and f is the frequency that GB is broadcasting the timestamps.	177

6.4	Area covered by a beacon in height.	181
A.1	Hancke-Kuhn Protocol [HK05]. $f_x()$ is a pseudo random function.	203
A.2	Reid <i>et al.</i> distance-bounding protocol [RNTS07]. $f_x()$ is a pseudo random function.	204
A.3	TDB Protocol [ALM11]	205
A.4	TDB Protocol with PRF Masking [BMV12]	206
A.5	Mutual Authenticated Distance Bounding protocol (MAD) [ČBH03]	208
A.6	Mutual Authenticated Distance Bounding protocol [SP07a]	209
A.7	The verifier measures the time between sending a challenge signal $c(t)$ and receiving the reply signal $r(t) = r_1(t) + r_2(t)$. If $c(t) = r(t)$, the distance bound to the prover is then given by $(t_r - t_0) \cdot c$, where c is the speed of light. [RČ08]	213
A.8	Schematic of prover by using CRCs [RČ08]	214
A.9	Overview of the switched challenge reflector with carrier shifting [RTŠ ⁺ 12]	215
A.10	The channel shifter. The incoming signal $c'(t)$ contains the challenges on either carrier frequency w_0 or w_1 . After mixing $c(t)$ with w_Δ , the signal is filtered appropriately to generate the four possible response channels: $w_0 - w_\Delta$, $w_0 + w_\Delta$, $w_1 - w_\Delta$, $w_1 + w_\Delta$. [RTŠ ⁺ 12]	215
A.11	Switched channel activator. The registers R^0 and R^1 , which are derived from the key of prover, select which two of the four reply channels are used in this round. The channel in which sufficient energy is encountered first gets enabled. After a channel is activated, it stays active until the end of this rapid bit-exchange round while the other channels remain deactivated until the end of this round. [RTŠ ⁺ 12]	216
A.12	Internals of channel activation. We obtain a DC component of the squared signal to detect energy in the channel and store the value for this round in a latch-like circuit. In summary, the “Energy Detector” returns 1 if there is a signal on the input, and 0 otherwise. The channel activation can be disabled by pulling EN (enable signal) low and is automatically reset at the beginning of each round of the rapid-bit exchange (RST). [RTŠ ⁺ 12]	216

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
$\{0, 1\}^*$	binary string
$\{0, 1\}^L$	binary string of length $L \in \mathbb{N}$
1^λ	security parameter of length $\lambda \in \mathbb{N}$
\mathcal{V}	set of verifiers
\mathcal{U}	set of users, each identifiable with a secret certificate
\mathcal{P}	set of provers
\mathcal{P}^i	provers subset user i in \mathcal{U}
\mathcal{T}	set of actors
\mathcal{X}	set of all participants in an experiment
\mathcal{X}^*	set of corrupted participants
\mathcal{B}	distance upper bound used in a DB protocol
\mathcal{C}	speed of light; $\mathcal{C} \approx 3 * 10^8$ meters per second
GM	group manager
GB	group beacon
Π	DB protocol between the prover and the verifier
$\Pi.V$	algorithm of the honest verifier in the Π protocol
$\Pi.P$	algorithm of the honest prover in the Π protocol
Out_V	output of the verifier a Π protocol
p_{noise}	noise probability of the fast phase of DB protocol
$stTime(\Gamma)$	execution start time of the protocol Γ
$fshTime(\Gamma)$	execution finish time of the protocol Γ
$exTime(\Gamma)$	time range from $stTime(\Gamma)$ to $fshTime(\Gamma)$
$exLen(\Gamma)$	execution length = $fshTime(\Gamma) - stTime(\Gamma)$

$View_x^\Gamma(e)$	view of x right after the event e in the protocol Γ
$View_x^\Gamma$	short for $View_x^\Gamma(e_{last})$, where e_{last} is the last protocol event
loc_x	location of participant $x \in \mathcal{X}$
$d(a,b)$	distance between the two locations a and b

Chapter 1

Introduction

Authentication is a prerequisite of any access control system. During an authentication protocol, a prover must prove to the verifier that it has the correct credentials. The verifier will either accept the prover as being legitimate, or reject if it is illegitimate. Authentication systems are considered secure if they prevent impersonation attacks, in other words, an illegitimate prover must not succeed in authenticating itself to the verifier.

In some authentication scenarios, the distance between the verifier and prover needs to be less than a certain threshold. For example, the customers of a grocery store should be located close to the payment terminal for making the transaction with their credit card. The security models of classic authentication do not capture man-in-the-middle (MiM) relay attacks, where an adversary just forwards data between the prover and the verifier. In a MiM attack the adversary, who is located within distance bound \mathbb{D} to the verifier, commits the fraud and convinces the verifier that the legitimate prover is located within the distance bound to the verifier, and as a result it can relay the messages between the payment terminal and the credit card of a victim. Figure 1.1 and Figure 1.2 show the difference between a legitimate and fraudulent access to a confidential area that is controlled by a door.

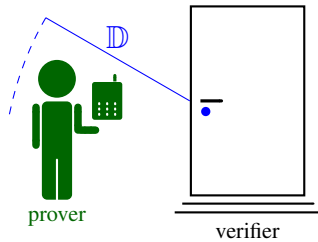


Figure 1.1: Legitimate Access

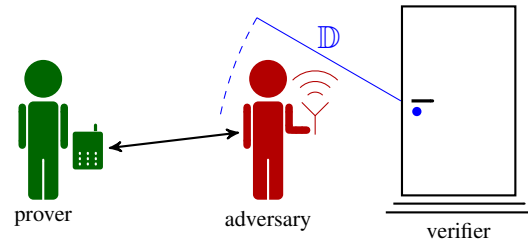


Figure 1.2: Fraudulent Access

Distance bounding (DB) protocols were first proposed in [Des88] to provide security against Man-

in- the-middle (MiM) attack in authentication protocols, and later found wide applications in location and proximity based services [BMV13b, RC10, FDC11, HPVP11, CRSC12]. All of the existing secure DB protocols use the traveling time of messages to measure the distance between the verifier and the prover. Figure 1.3 shows a generic DB protocol that includes "slow" and "fast" phases. The fundamental difference between slow and fast phase is that in the fast phase, the verifier is measuring the time between the challenge and response messages, while there is no time measurement in slow phase. In the first slow phase, a table is created by the prover (sometimes the verifier calculates the table too). In the fast phase, the verifier asks for some values on the table. And in the last slow phase, the verifier checks the validity of responses and the time measurements.

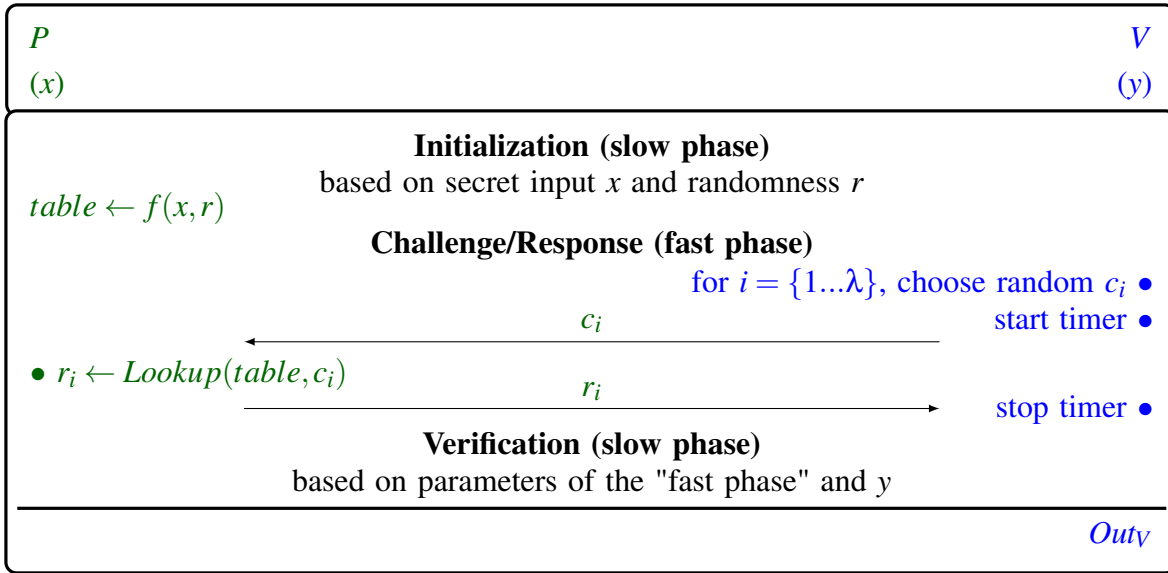


Figure 1.3: Generic DB protocol. In shared-key setting, $y = x$.

Most DB protocols are symmetric key protocols where the prover and the verifier share a secret key. One of the main concerns in using these protocols in real life applications is privacy of the user's location information. More recently public-key DB protocols, where the prover has a registered public-key, have been proposed [ASN14, Vau14, ASN17a]. In [ASN14] it is shown how to extend the protocol to provide user anonymity, in the sense that transcripts of protocol execution cannot be linked to a user, nor to any other transcript. This is a very attractive property that allows user to enjoy location based services while their privacy is maintained. While much research has

been reported on formalizing security of DB protocols and designing protocols that have provable properties [DFKO11, BMV13a], formalizing the security of the anonymous DB protocols is less understood.

In a DB setting there are three types of *participants*: *provers* who are registered in the system and have secret keys, a *verifier* who is honest and has access to correct public keys of provers, and *actors* who are not registered in the system, but want to be accepted and may be in collusion with a dishonest prover.

In a symmetric setting, the secret of the prover is shared with the verifier and so the challenge/response table can be calculated by the verifier too. In a public-key setting, the verifier only has access to the public-key of the prover, which does not allow them to calculate the challenge/response table. In this case, the verifier checks the correctness of the responses of the prover using the relation between them and the prover's public-key.

For a DB protocol with distance bound \mathcal{B} , we refer to participants whose distance to the verifier is less than \mathcal{B} as "close-by" participants and those who are farther away than \mathcal{B} , as "far-away" participants.

Important attacks against DB protocols are;

1. *Distance-Fraud* [BC94]; where a dishonest far-away prover tries to be accepted in the protocol.
2. *Mafia-Fraud (MF)* [Des88]; a close-by actor tries to use the communications of a far-away honest prover, to succeed in the protocol.
3. *Terrorist-Fraud (TF)* [Des88]; a dishonest far-away prover colludes with a close-by actor, according to succeed in the protocol.

Most of the existing DB protocols are in shared-key setting, where any pair of verifier and prover is assigned with a unique shared-key. In these models, the verifiers are assumed to be trusted and do

not deviate from the intended purpose.

The verifier algorithm is used by a service provider that provides some functionality in the system, such as opening a door or make a payment. In order to have security guarantee, the service provider needs to have tight control on the verifier devices and their communication channels, because the shared-keys (the only asset of parties) are well used in the system and any successful attack on any of the verifier can break the whole system. This setting limits the applications of DB protocols. One of our general goals in this thesis is to design models and protocols that require less control on the verifier, such as public-key and anonymous DB.

In most of existing DB protocols we have shared-key between prover and verifier. Although these protocols are relatively simple, but there are key management problems and limitations in the security models. As an example, we mention the following limitations;

- The DB authentication can only run by entities who are allowed to have the secret shared-key. This makes the verifier as a target for the attacker, who is interested to gain the secret shared-keys in possession of the verifier.
- Since the verifier should have the shared-key, there will be no privacy for prover against the verifier.

Some authentication models need to overcome these limitations. For example in anonymous authentication, prover's privacy from the verifier is a requirement.

An alternative to the shared-key setting is public-key DB protocols in which provers have a long term key pair and they just publish the public-key to the verifiers. This approach leaves no incentive for an attacker to target the verifier. In this research we generally move towards models with less trusted verifiers. At the time this research began, there was no public-key DB protocol that is secure against all DB attacks.

Contributions. In this thesis, we consider three major topics: Public-Key DB, Anonymous DB, and

One-Shot DB. In public-key DB, input of the verifier is limited to the public-key of provers. In anonymous DB, input of the verifier is limited to the public parameters of the system and verifier must not identify the prover. In One-Shot DB, any two parties can form a prover and a verifier, and the verifier does not transmit any message. We discuss each of these three topics in a separate section.

In traditional symmetric key DB models, as the prover and verifier share a piece of data (shared-key), it makes sense to define DB protocol as a special *authentication* scheme, which consists of *identification* of prover and *correctness* of the shared-key that is used by prover in the protocol, and the proof of distance. Some DB models design the protocol as a special *proof-of-knowledge* scheme [Vau14]. However in public-key and anonymous DB schemes, since the verifier does not have the secret of the prover, we are just interested in the proof of the identity of prover (*identification*). We consider DB schemes as an *identification* scheme.

In the distance-bounding literature, it is assumed that all entities use omni-directional antennas for wireless communication. This is too strong an assumption given that today's communication protocols have started to consider directional antenna [ARS16]. We propose a new class of attacks that uses directional antenna to break the security guarantees of existing DB protocols. In fact, we show specific attacks against some public-key DB protocols and all existing anonymous DB protocols. We propose models that consider this type of attack and protocols that are secure in these models.

In all attacking scenarios of distance-bounding literature, the agents of at most one user are involved. In other words, the credentials of at most one user is used by the adversary. However in some attack scenarios, the collusion of multiple users can help the adversary to break a DB protocol with higher chance. In a collusion attack, multiple users (each with a secret key) participate in the attack that can be in DF, MF or TF form. Note that there is difference between two multi party attacks: 1. collusion between two legitimate provers (which is named as *collusion* attack) and, 2. collusion between a legitimate user and an actor (which is called TF attack).

Collusion attacks had not been considered before and need not be considered as long as the secret keys of two users are independently generated, because a protocol transcript (without anonymity) can be linked to a user through their key information and so cannot be combined with other transcripts to form a new forged transcript. In anonymous DB protocols however, the verifier should not be able to link the transcript of a protocol to a single user and so combining protocol transcripts can give advantage to colluders. We show that collusion TF attack can be used to subvert traceability functionality of the anonymous DB. This functionality is necessary in all anonymous DB protocols to ensure user accountability by allowing a third party that holds a master key, to “open” a transcript and identify the user, when required. We propose models that consider collusion attack and protocols that are secure in these models.

1.1 Public-Key Distance-Bounding

Considering public-key setting rather than existing symmetric key DB protocols is a step towards having less trusted verifiers. In a public-key setting, the verifier only has access to the public keys of provers.

There are only a few public-key DB protocols in the literature; The seminal DB protocol [BC94], which is in fact a public-key DB protocol, is not *TF* resistant. Bussard-Bagga [BB04] protocol, which was designed to provide *TF* resistance, is shown to be vulnerable against a *TF* attack [BBM⁺13]. Hermans *et al.* [HPO13] protocol is not *TF* resistant. Vaudenay [Vau14] proposed the first formal model for public-key DB, which is based on proof-of-knowledge schemes. Vaudenay proposed a protocol that is secure against all DB attacks.

A challenging problem in public-key DB is having a model that is suitable for the type of attacks in this literature. More specifically, in DB literature the MF attack (Attack 3.1.2) is a type of man-in-the-middle attack that needs to be considered in the model. The proposed model in Vaudenay [Vau14] is an extension of proof-of-knowledge schemes (Definition 2.5.3), which does not natu-

rally consider MiM attacks. We elaborate more in Chapter 4.

Another challenge in public-key DB compared to symmetric key DB is coming from the fact that verifier does not know the private-key of the prover. The fast phase operations usually take place on individual bits of prover's private-key, however the verifier has access to the public-key of prover, which is normally a mathematical operation on the whole private-key. The validation of every single bit of the fast phase is a challenging task for the verifier, while in symmetric key DB the verifier has access to every single bit of the shared-key, which makes validation of every bit of the fast phase way easier.

We propose a formal model of public-key DB scheme that is inline with cryptographic identification schemes (Definition 2.4.1), and propose different constructions that are secure against all DB attacking scenarios. Each construction is using a specific well known public-key setting. The diversity of public-key setting allows us to pair the DB schemes with different public-key systems.

1.2 Anonymous Distance-Bounding

In some authentication models, the provers need to have some level of privacy against the verifier. The privacy of a prover can be either about hiding the behavior, or the identity of the prover from the verifier. For example, in anonymous authentication models, although the prover needs to prove its authenticity to the verifier, but the verifier does not learn the identity of the prover.

In authentication schemes that involve the location of provers, the privacy problem is either about hiding the location behavior or the identity of the prover. In order to hide the location behavior, the prover can modify its location data or provide inaccurate location data [Kru09, STLBH11]. However, in many location based application, accurate location information is required by the verifier. In order to achieve both privacy and location accuracy, we can unlink the location data from the identity of provers. Anonymous DB allows the verifier to have verifiable distance guarantee, while the identity of the prover is hidden from the verifier.

In another class of studies, [Vau07, HPVP11] considered privacy against eavesdroppers or active third parties, other than provers and verifiers. This problem has been studied in the presence of MiM adversary, trusted verifiers and trusted registration authority. However, hiding the identity of the prover from the verifier is yet an open problem.

Designing an anonymous DB scheme has extra challenges compared to public-key DB, because not even the prover has to anonymously prove its membership to the verifier, but also the verifier has to check that every single bit of the fast phase is involving a bit of the private certificate of the prover. We elaborate more in Chapter 5.

We propose the first formal model of anonymous DB scheme inline with anonymous group identification (Definition 2.4.5), and propose different constructions that are secure against all DB attacking scenarios. Each construction is using a specific well known public-key setting. The first anonymous DB scheme of the literature that is secure against all DB is among these constructions. We also propose a protocol that converts any public-key DB scheme that use a few public-key settings to an anonymous DB scheme.

1.3 One-Shot Distance-Bounding

All existing distance-bounding systems consider some assumptions that are easily violated in real scenarios. This fact dramatically limits the applications of this field. Here we mention some of these assumptions;

Mobile Participants. In all existing distance-bounding models, it is assumed that the parties are not moving during the execution of the protocol. However in some scenarios, the two parties need to authenticate while they are moving. A good example of an application is ad-hoc network applications that use distance-bounding (See Figure 1.4).

The non-mobile assumption of parties is because of the time of travel technique that is used in

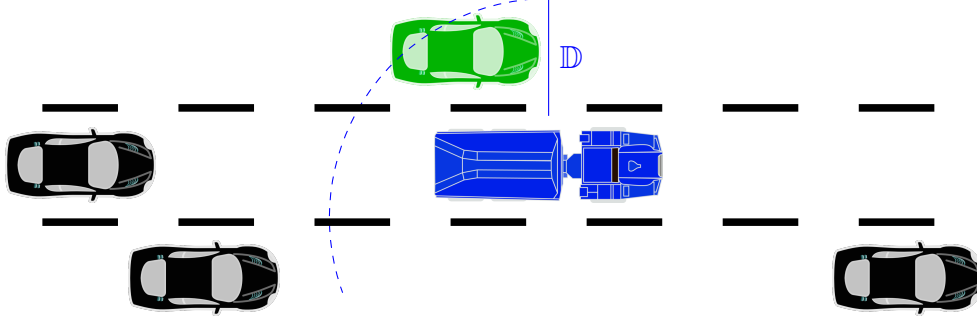


Figure 1.4: Mobile participants in DB

the existing DB protocols. In order to measure the distance of a prover, as it is shown in the fast phase of Figure 1.3, the verifier sends a challenge message that travels with speed of light. Upon receiving the challenge message, the prover sends a response message. The verifier finds the mutual distance by measuring the send/receive time of messages. However if the parties are moving, the timing of each message will be different in each round, which causes measurement inaccuracy.

Scalable and Continuous DB. As it is shown in Figure 1.3, a DB protocol includes multiple fast phase challenge-response rounds, which is quite costly. The communications of the fast phase are non-reliable, for example the existing implementations are suggesting use of silent channel frequency range for each possible fast phase message [RTŠ⁺12]. Therefore, in order to run one DB session, a few silent channels (4 channels in [RTŠ⁺12]) has to be maintained during the whole session.

In a populated area with many provers, this leads to either congestion problem or very high cost. This problem prevents having an scalable DB solution. Moreover, in some DB scenarios the prover needs to prove the proximity continuously over time, for example in key-less entry car scenario the prover needs to be always in the car for engine to work. We have the same cost problem in this scenario, which prevents having continuous DB solution.

Implementation Difficulties. One of the main challenges in distance-bounding protocols, that are secure against all DB attacks, is their implementation. Since distance measurement is done based on speed of light, the processing time for responding to the challenges should be comparable to the propagation time of signal. Unfortunately achieving this requirement is very hard.

There are just a few implementations of DB protocols in the literature [RČ08, Tip12, RTŠ⁺12], each making specific hardware for the process of fast phase challenge-response at the prover. In all of them the verifiers need to send a fast challenge to the prover, and the prover needs to compute the proper response based on the challenge. Among these implementations, only one is secure against all three of the mentioned attacks [RTŠ⁺12]. And even this implementation is enforcing some response delay that makes the protocol a few meters inaccurate in all distance-bounding attacks.

Our Solution. In order to solve most of these problems, for the first time, we propose a synchronous time model that considers the location of participants in authentication protocols. Based on this time model, we propose the first One-Shot distance-bounding model. Despite all the existing secure distance-bounding models that use the challenge-response technique to measure the distance between the participants, in One-Shot DB it is only the prover that sends a single message and the verifier is passive in the protocol. This reduces the number of fast phase messages from 2ℓ , ℓ being the number of fast phase rounds, to just one message.

This drastic reduction in the communication cost, allows us to have more scalable system. Moreover, it allows the verifier to check the proximity of a single prover more often in the cases where continuous presence of the prover is required, rather than the one-time proximity verification of typical DB protocols. The location of the prover is only considered at one moment that is the sending time of the message. In this model, the participants can move and yet the location measurements are accurate as long as the parties move rather slower than the speed of light.

We propose a One-Shot DB constructions that is secure against all DB attacks. In this approach, we consider a global clock that facilitates synchronization of participants, which allows us to build the

DB protocol. In this approach we still have implementation challenges.

1.4 Thesis Organization

In Chapter 1 we introduced the problems that we are looking in this thesis. In Chapter 2 we describe the background and notations that we use throughout the thesis. In Chapter 3 we review the more related works to the main concepts of the thesis, and leave the less related works for Appendix A. Our contributions are presented in three chapters: Chapter 4 contains public-key distance-bounding, Chapter 5 contains anonymous DB, and Chapter 6 contains one-shot DB. Chapter 7 concludes the thesis.

Chapter 2

Background and Notations

In this chapter, we recall some of the background on cryptographic primitives and mathematical theorems that will be used throughout this thesis.

The notations that we use in this thesis follows the following standard: sets are denoted by the font \mathbb{N} . Operations are decision problems that are denoted by the font Alg , such as algorithms, protocols and functions. Security parameter (λ) of an operation is a variable that measures the input size of the computational problem. We denote the security parameter of an operation represented in unary, as input $\text{Alg}(1^\lambda)$. The output of an operation is denotes by $out \leftarrow \text{Alg}$. Probabilistic Polynomial Time (PPT) is the class of decision problems (or operations) solvable by a probabilistic Turing machine in polynomial time. Considering Σ as a non-empty finite set of symbols, called the alphabet. Σ^* denotes a string from alphabet Σ that is a repeated selection from Σ , for example $\{0, 1\}^*$ denotes a binary string that is any consecutive binary text. Participant sets are denoted by the font \mathcal{P} . *Language* L is a set of strings of symbols together with a set of rules that are specific to it.

The general security goals in this thesis are as follows; **Confidentiality:** Eve can eavesdrop on the channel and get messages, but she cannot do the same computation as intended receiver, as she does not know the key of intended receiver. **Authentication:** When Bob receives a message from Alice, he knows that it is really Alice who sent the message. **Integrity:** Bob obtains assurance that the message sent by Alice has not been altered by a third party. **Non-repudiation:** After Alice sends a message to Bob, she cannot claim that she has not send it.

2.1 Encryption

We use encryption scheme in order to provide confidential communication as follows: Alice and Bob agree on a secret key that they must share by means of a secure channel. At the end of key agreement, Alice gets $e \in \mathbb{K}$ and Bob gets $d \in \mathbb{K}$. Once the keys has been agreed on, Alice takes the plaintext p and computes the ciphertext $c = \text{Enc}_e(p)$ which then sends to Bob. Upon receiving c , Bob computes $p = \text{Dec}_d(c) = \text{Dec}_d(\text{Enc}_e(p))$, recovering the plaintext.

Here we formally describe encryption schemes;

Definition 2.1.1 ((Encryption Scheme) [Buc04]) *An encryption scheme is defined as a tuple $(\mathbb{P}, \mathbb{C}, \mathbb{K}, \mathbb{E}, \mathbb{D}, \text{KeyGen}, \text{Enc}, \text{Dec})$ as follows;*

1. \mathbb{P} is a set called the "plaintext space". Its elements are called plaintexts.
2. \mathbb{C} is a set called the "ciphertext space". Its elements are called ciphertexts.
3. \mathbb{K} is a set called the "key space". Its elements are called keys. The algorithm that picks one or more keys is called "key generator" ($\text{KeyGen}(1^\lambda)$).
4. $\mathbb{E} = \{\text{Enc}_k : k \in \mathbb{K}\}$ is a set of functions $\text{Enc}_k : \mathbb{P} \rightarrow \mathbb{C}$. Its elements are called encryption functions.
5. $\mathbb{D} = \{\text{Dec}_k : k \in \mathbb{K}\}$ is a set of functions $\text{Dec}_k : \mathbb{C} \rightarrow \mathbb{P}$. Its elements are called decryption functions.

The properties of an encryption scheme is as follows:

- **Correctness:** $\Pr[(e, d) \leftarrow \text{KeyGen}, c = \text{Enc}_e(p), p = \text{Dec}_d(c)] = 1$ for all $p \in \mathbb{P}$.
- **Security:** $\forall p_1, p_2 \in \mathbb{P}, \{(e, d) \leftarrow \text{KeyGen}, c_1 = \text{Enc}_e(p_1) : c_1\} = \{(e, d) \leftarrow \text{KeyGen}, c_2 = \text{Enc}_e(p_2) : c_2\}$, where the first formula refers to the distribution of all cipher texts c_1 reached by randomly choosing key (e, d) and encrypting message p_1 , the second is the same but for p_2 . A different security definition will be provided in Definition 2.1.2.

This definition often gets modified in order to distinguish an encryption scheme as being either a symmetric-key or public-key encryption scheme. In **symmetric-key** encryption the same key is used both for encryption and decryption functions. In **public-key** encryption however, the KeyGen generates a key pair (sk, pk) , where sk is known as private key and pk is known as public key. The encryption function uses the public key as input (Enc_{pk}) and the decryption function uses the private key as input (Dec_{sk}).

The security property of encryption schemes is defined in different ways. The basic approach of Definition 2.1.1 in definition security is called "perfect security". However, in this thesis we are interested in computational definitions of security. A well-accepted approach is defining a distinguishability game to define computational security. Intuitively, if an encryption scheme possesses the property of indistinguishability, then an adversary will be unable to distinguish pairs of ciphertexts based on the message they encrypt. The property of indistinguishability under chosen plaintext attack (IND-CPA) is considered a basic requirement for most provably secure public key encryption, though some schemes also provide indistinguishability under chosen ciphertext attack (IND-CCA1) and adaptive chosen ciphertext attack (IND-CCA2). In the following we describe these properties in public-key encryption.

The classical goal of secure encryption is to preserve the privacy of messages: an adversary should not be able to learn from a ciphertext information about its plaintext beyond the length of that plaintext. A version of this notion, indistinguishability of encryptions (IND) is defined in [BDPR98] through an experiment: Algorithm \mathcal{A}_1 is run on input the public key, pk . At the end of \mathcal{A}_1 's execution she outputs a triple (p_0, p_1, s) , the first two components being messages with the same length, and the last being state information (possibly including pk) which she wants to preserve. The challenger randomly selects $b \in_R \{0, 1\}$, and returns $c = \text{Enc}_{pk}(p_b)$ to adversary \mathcal{A}_2 . It is \mathcal{A}_2 's job to determine the bit b . To make this determination \mathcal{A}_2 is given the saved state s and the challenge ciphertext c .

We simultaneously define indistinguishability with respect to CPA, CCA1 and CCA2. The only

difference lies in whether or not \mathcal{A}_1 and \mathcal{A}_2 are given access to decryption oracles. We let the string atk be replaced by any of the formal symbols $cpa, cca1, cca2$, while ATK is then the corresponding formal symbol from CPA, CCA1, CCA2. The notation $\mathcal{O}_i = \perp$, for $i \in \{1, 2\}$, indicates that \mathcal{O}_i is the function which, on any input, returns the empty string \perp .

Definition 2.1.2 (Indistinguishable Security [KL14]) *Let $\Pi = (\mathbb{P}, \mathbb{C}, \mathbb{K}, \mathbb{E}, \mathbb{D}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme according to Definition 2.1.1, where $(sk, pk) \leftarrow \text{KeyGen}(1^\lambda)$ and encryption/decryption functions take public/private keys respectively $(\text{Enc}_{pk}/\text{Dec}_{sk})$. And let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be the adversary. For $atk \in \{cpa, cca1, cca2\}$ and $\lambda \in \mathbb{N}$ we define*

$$\text{Adv}_{\mathcal{A}, \Pi}^{\text{ind-atk}}(\lambda) := 2\Pr[b \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(p_0, p_1, s, c) | \text{event}_1] - 1, \text{ where } \text{event}_1 \text{ is defined as}$$

$$\text{event}_1 := (pk, sk) \leftarrow \text{KeyGen}(1^\lambda); (p_0, p_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk); b \leftarrow_R \{0, 1\}; c = \text{Enc}_{pk}(p_b)$$

The adversaries \mathcal{A}_1 and \mathcal{A}_2 have oracle access to \mathcal{O}_1 and \mathcal{O}_2 respectively as follows:

$$\begin{aligned} \text{if } atk = cpa, \quad \text{then } \mathcal{O}_1(\cdot) = \perp \quad \text{and } \mathcal{O}_2(\cdot) = \perp \\ \text{if } atk = cca1, \quad \text{then } \mathcal{O}_1(\cdot) = \text{Dec}_{sk}(\cdot) \quad \text{and } \mathcal{O}_2(\cdot) = \perp \\ \text{if } atk = cca2, \quad \text{then } \mathcal{O}_1(\cdot) = \text{Dec}_{sk}(\cdot) \quad \text{and } \mathcal{O}_2(\cdot) = \text{Dec}_{sk}(\cdot) \end{aligned}$$

In the case of CCA2, \mathcal{A}_2 does not ask its oracle to decrypt c . We say that Π is secure in the sense of IND-ATK if for any polynomial-time \mathcal{A} , $\text{Adv}_{\mathcal{A}, \Pi}^{\text{ind-atk}}(\lambda)$ is negligible in terms of λ .

2.2 Signature

Another cryptographic primitive that we use is digital signature. Signature scheme is used for demonstrating the authenticity of a messages. A valid signature gives a recipient reason to believe that the message was created by a known sender (**authentication**), that the sender cannot deny having sent the message (**non-repudiation**), and that the message was not altered in transit (**integrity**).

Definition 2.2.1 (Signature Scheme [Gol09]) *A signature scheme is defined as a tuple $(\mathbb{T}, \mathbb{PK},$*

$\mathbb{SK}, \mathbb{S}, \mathbb{V}, \text{KeyGen}, \text{Sign}, \text{Verify}$) as follows;

- (I) \mathbb{T} is a set called the "tag space". Its elements are called signature tags.
- (II) \mathbb{PK} and \mathbb{SK} are the sets of "public-key space" and "private-key space", and their elements are called public-keys and private-keys respectively. The algorithm that picks a public-key (pk) and the corresponding private-key (sk) is called "key generator" ($(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$).
- (III) $\mathbb{S} = \{\text{Sign}_{sk} : sk \in \mathbb{SK}\}$ is a set of functions $\text{Sign}_{sk} : \{0, 1\}^* \rightarrow \mathbb{T}$. Its elements are called sign functions, that generate a signature tag for any message.
- (IV) $\mathbb{V} = \{\text{Verify}_{pk} : pk \in \mathbb{PK}\}$ is a set of functions $\text{Verify}_{pk} : \{0, 1\}^* \times \mathbb{T} \rightarrow \{\text{accept}, \text{reject}\}$. Its elements are called verify functions that validate a signature tag corresponding to a message.

The properties of a signature scheme is as follows:

- **Correctness:** $\Pr[(pk, sk) \leftarrow \text{KeyGen}(1^\lambda), \text{Verify}_{pk}(x, \text{Sign}_{sk}(x)) = \text{accept}] = 1$.
- **Security:** A signature scheme is secure if for every PPT adversary \mathcal{A} we have

$$\Pr[(pk, sk) \leftarrow \text{KeyGen}(1^\lambda); (x, t) \leftarrow \mathcal{A}^{\mathcal{O}^{\text{Sign}_{sk}(\cdot)}}(pk, 1^\lambda); x \notin \mathbb{Q}; \text{Verify}_{pk}(x, t) = \text{accept}] < \text{negl}(\lambda),$$

where $\mathcal{O}^{\text{Sign}_{sk}(\cdot)}$ denotes that \mathcal{A} has oracle access to $\text{Sign}_{sk}(\cdot)$ function, and \mathbb{Q} denotes the set of queries on $\mathcal{O}^{\text{Sign}_{sk}(\cdot)}$ that already made by \mathcal{A} . Note that the adversary cannot directly query the string x on $\mathcal{O}^{\text{Sign}_{sk}(\cdot)}$.

2.3 Authentication Protocol

Unfortunately there is no universally accepted definition of entity authentication, so here we describe the most accepted definition. Bellare-Rogaway [BR93] proposed the first formal model of security for the analysis of entity authentication protocols. It is a game-based definition, in which

the adversary is allowed to interact with a set of oracles that model communicating parties in a network.

Let oracle $\pi_{i,j}^s$, $i, j \in I$ denote player i (called initiator) attempting to authenticate to player j (called responder) in session s , where $I \subseteq \{0, 1\}^k$ is the set of identities that defines players. The adversary \mathcal{A} is a probabilistic machine equipped with an infinite collection of oracles $\pi_{i,j}^s$, $i, j \in I$ and $s \in \mathbb{N}$. Adversary \mathcal{A} communicates with the oracles via queries of the form (i, j, s, x) written on a special tape. The query is intended to mean that \mathcal{A} is sending message x to i , claiming it is from j in session s . The query will, in our model, be answered by $\pi_{i,j}^s$ using the following experiment.

Definition 2.3.1 (BR Protocol) *Running a protocol Π (with long-lived key generator \mathcal{G}) in the presence of an adversary \mathcal{A} , using security parameter k , means performing the following:*

1. Choose a random string $r_G \in \{0, 1\}^*$, set $a_i = \mathcal{G}(1^k, i, r_G)$ for $i \in I$, and set $a_{\mathcal{A}} = (1^k, \mathcal{A}, r_G)$.
2. Choose a random string $r_{\mathcal{A}} \in \{0, 1\}^*$, and for each $i, j \in I$ and $s \in \mathbb{N}$ choose a random string $r_{i,j}^s \in \{0, 1\}^*$.
3. Let $\kappa_{i,j}^s = \emptyset$ for all $i, j \in I$ and $s \in \mathbb{N}$. This variable keeps track of the conversation that $\Pi_{i,j}^s$ engages in.
4. Run adversary \mathcal{A} on input $(1^k, a_{\mathcal{A}}, r_{\mathcal{A}})$ answering oracle calls as follows: When \mathcal{A} asks a query (i, j, s, x) oracle $\Pi_{i,j}^s$ computes $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa_{i,j}^s || x, r_{i,j}^s)$ and answers with (m, δ) . Then $\kappa_{i,j}^s$ gets updated to $\kappa_{i,j}^s || x$.

We point out that in response to an oracle call, \mathcal{A} learns not only the outgoing message but also whether or not the oracle has accepted or rejected.

An adversary is called *benign* if it is deterministic and restricts its action to choosing a pair of oracles $\pi_{i,j}^s$ and $\pi_{j,i}^t$ and then faithfully conveying each flow from one oracle to the other, with $\pi_{i,j}^s$ beginning first. In other words, the first query \mathcal{A} makes is (i, j, s, λ) , generating response α_1 ; the second query \mathcal{A} makes is (j, i, t, α_1) generating response β_1 and so forth. In the presence of a

benign adversary, $\pi_{i,j}^s$ has a matching conversation to $\pi_{j,i}^t$, that will be defined in Definition 2.3.2;

Let $T^{i,s}$ denote all messages sent and received by $\pi_{i,j}^s$ in chronological order. $T^{i,s}$ is known as the transcript of $\pi_{i,j}^s$. For two transcripts $T^{i,s}$ and $T^{j,t}$, we say that $T^{i,s}$ is a prefix of $T^{j,t}$, if $T^{i,s}$ contains at least one message, and the messages in $T^{i,s}$ are identical to and in the same order as the first $|T^{i,s}|$ messages of $T^{j,t}$.

Definition 2.3.2 (Matching Conversation) $\pi_{i,j}^s$ has a matching conversation to $\pi_{j,i}^t$, if

- $T^{j,t}$ is a prefix of $T^{i,s}$ and $\pi_{i,j}^s$ has sent the last message(s), or
- $T^{i,s} = T^{j,t}$ and $\pi_{j,i}^t$ has sent the last message(s).

Two processes $\pi_{i,j}^s$ and $\pi_{j,i}^t$ have matching conversations if $\pi_{i,j}^s$ has a matching conversation to process $\pi_{j,i}^t$, and vice versa.

Now we are ready to define secure authentication, which is defined in a way that captures both one-way and mutual authentication;

Definition 2.3.3 (Secure Authentication) Let $\text{NoMatching}^{\mathcal{A}}(\mathbf{k})$ denote the event that, when protocol P is run against adversary \mathcal{A} , there exists an oracle $\pi_{i,j}^s$ with $i, j \notin \hat{\mathcal{A}}$ (where $\hat{\mathcal{A}}$ denotes the set of entities corrupted by \mathcal{A}) which accepted but there is no oracle $\pi_{j,i}^t$ which has a matching conversation to $\pi_{i,j}^s$.

A protocol P is a secure (mutual) authentication protocol if for every adversary \mathcal{A} :

1. **(matching conversation \Rightarrow acceptance)** if $\pi_{i,j}^s$ and $\pi_{j,i}^t$ have matching conversations, then (both oracles) accept.
2. **(acceptance \Rightarrow matching conversation)** the probability of $\text{NoMatching}^{\mathcal{A}}(\mathbf{k})$ is negligible.

The first condition says that if each party's messages are faithfully relayed to one another, then the parties accept the authentication of one another. The second condition calls an execution good if

for each accepting conversation K by an oracle $\pi_{i,j}^s$ there exists a matching conversation K' by some oracle $\pi_{j,i}^t$, and otherwise it is a bad execution, which is required to have negligible probability.

2.4 Identification

Identification scheme (ID) is a cryptographic component that allows a prover P to convince a verifier V that they know a witness x related to a public value z . Identification scheme guarantees that a specific entity is involved in the protocol. The integrity of messages is not necessarily required in identification, but the entity authentication consists of both identification and message integrity. In the following, we describe a well known definition about identification schemes.

Definition 2.4.1 (Identification Scheme [KH06]) *Identification scheme (ID) is an interactive protocol $(P(\zeta), V(z))$ of PPT algorithms operating on a language L and relation $\mathbb{R} = \{(z, \zeta) : z \in L, \zeta \in W(z)\}$, where $W(z)$ is the set of all witnesses for z . Prior running this protocol, the inputs of parties, (ζ, z) , are generated by KeyGen function. In this scheme, the prover convinces the verifier that it knows a witness in $W(z)$. This properties of identification scheme is as follows:*

- **Completeness:** $\Pr[Out_V = 1 : P(\zeta) \leftrightarrow V(z)] = 1$ for all $(z, \zeta) \in \mathbb{R}$, when P and V are honest.
- **κ -Soundness:** $\Pr[Out_V = 1 : P^* \leftrightarrow V(z)] \leq \kappa$ in any of the following two cases; (i) $z \notin L$, (ii) $z \in L$ while algorithm P^* is independent from any $\zeta \in W(z)$.

Another important property that was considered later for identification schemes, is man-in-the-middle resistance property;

Definition 2.4.2 (Identification MiM [LM13]) *Consider the following two polynomial adversaries in an identification scheme as in Definition 2.4.1: \mathcal{A}_1 interacts simultaneously with $P(\zeta)$ and $V(z)$ for polynomial number of times, and \mathcal{A}_2 only interacts with $V(z)$ by having some input as \mathcal{A}_1 .*

- **MiM resistant:** the success chance of any \mathcal{A}_2 , given the final view of any \mathcal{A}_1 , is negli-

ble. In other words, $\Pr[Out_V = 1 : \mathcal{A}_2(View_{\mathcal{A}_1}) \leftrightarrow V(z)] \leq \text{negl}$.

A well-accepted framework for defining identification schemes, is using Σ -Protocol. A Σ -Protocol is a 3-round cryptographic protocol between a prover P and a verifier V , in which the two parties interact, and at the end of the protocol, V is convinced about validity of P 's statement. P has a private input x that satisfies the relation $(x, y) \in \mathbb{R}$, where y is a public value that is also known to V .

Σ -Protocol [KH06] is defined as follows;

Definition 2.4.3 (Σ -Protocol) *A prover P and verifier V execute three algorithms (Commit, Response, Check) using inputs (x, y) and (y) , respectively in the following order. x is private and y is public.*

Let \mathbb{C} , \mathbb{H} and \mathbb{R} denote three sets defined as follows. \mathbb{C} is the set of possible input that is chosen by the prover; \mathbb{H} is the set of possible challenges chosen by the verifier; and \mathbb{R} is the set of possible responses of the prover. The steps of the protocol are as follows:

1. *P randomly chooses $a \in \mathbb{C}$, computes the commitment $A = \text{Commit}(a)$, and sends A to V .*
2. *Challenge and Response messages that are defined as follows:*
 - (a) *V randomly chooses a challenge $c \in \mathbb{H}$ and sends it to P ,*
 - (b) *P computes $r = \text{Response}(x, a, c) \in \mathbb{R}$ and sends it to V ,*
3. *V calculates $ret = \text{Check}(y, c, r, A)$, where $ret \in \{\text{accept}, \text{reject}\}$.*

At the end of the protocol, V outputs $Out_V = 1$ if $ret = \text{accept}$, and $Out_V = 0$ otherwise.

For implementing an identification scheme (Definition 2.4.1), we have $y = z$. We define an identification scheme in Σ -protocol framework as a tuple $ID = (\text{KeyGen}; \text{Commit}; \text{Response}; \text{Check})$. KeyGen is a PPT algorithm that generates (x, z) . The PPT algorithms Commit , Response and Check specifying an interactive protocol between the prover P and the verifier V as a Σ -protocol (Definition 2.4.3).

The properties are rephrased as follows: an identification scheme is complete if the Check function outputs *accept* if $(x, z) \in \mathbb{R}$ holds, and *reject* otherwise. An identification scheme is sound if an adversary with access to a set of valid transcripts $T = \{(A, c, r)\}$, cannot generate a valid transcript (A', c', r') for a c' that has not appeared in T . In another definition [KH06], the identification scheme has soundness when it is hard to compute two valid transcripts (A, c, r) and (A, c', r') such that $c \neq c'$. Note that a transcript (A, c, r) is valid according to the public key z , when the function $\text{Check}(z, c, r, A)$ returns *accept*.

Here, we propose an extended form of Σ -Protocols, called Σ^* -Protocols, in which the verifier consecutively sends multiple challenges, each (except the first one) after receiving the response to previous challenges. This form of protocol is needed in formalizing distance-bounding protocols, since there is consecutive challenge-response rounds with small messages, *e.g.*, bit size messages.

Definition 2.4.4 (Σ^* -Protocol) *A prover P and verifier V participate in the following interactive protocol.*

Let \mathbb{C} , \mathbb{H} and \mathbb{R} denote three sets defined as follows. \mathbb{C} is the set of possible input that is chosen by the prover; \mathbb{H} is the set of possible challenges chosen by the verifier; and \mathbb{R} is the set of possible responses of the prover. The steps of the protocol are as follows:

1. *P randomly chooses $a \in \mathbb{C}$, computes the commitment $A = \text{Commit}(a)$, and sends A to V .*
2. *Challenge and Response messages that are defined as follows:*

(a) V randomly chooses a challenge $c \in \mathbb{H}$ and sends it to P ,

(b) P computes $r = \text{Response}(x, a, c, \vec{c}) \in \mathbb{R}$, where \vec{c} is the list of previous challenges before c , and sends it to V ,

Steps 2-(a) and 2-(b) may be repeated a number of times.

3. *V calculates $\text{ret} = \text{Check}(y, [c], [r], A)$, where $\text{ret} \in \{\text{accept}, \text{reject}\}$ and $[c]$ and $[r]$ are lists of all challenges and responses, respectively.*

At the end of the protocol, V outputs $Out_V = 1$ if $ret = accept$, and $Out_V = 0$ otherwise.

Another notion of identification is when users can identify themselves anonymously as members of a group.

Definition 2.4.5 (Anonymous Group Identification [DDP06]) *anonymous identification scheme is defined by tuple $(KeyGen, Join, Prove)$, where $KeyGen$ is a PPT algorithm, and both $Join$ and $Prove$ are two party protocols of PPT algorithms. Initially, group manager runs $(sk, pk) \leftarrow KeyGen(1^\lambda)$ to get private-key sk and public-key pk such that they are in relation $\mathbb{R} = \{(sk, pk) : pk \in L, sk \in W(pk)\}$ where $W(pk)$ is a witness function that generates all witnesses of pk . A new user U , in order to join the system, executes $Join = (U(pk), GM(sk, pk))$ protocol with the group manager GM . Common input is pk and the private input to GM is sk . $Join$ protocol outputs to GM either "reject" or a string id , and outputs to U either "reject" or a membership certificate $cert_U$ such that it is in relation $\mathcal{R}' = \{(cert_U, pk) : pk \in L, cert_U \in W'(pk)\}$ where $W'(pk)$ is a witness function that generates all witnesses of pk . An already joined user P , in order to prove the membership, executes $Prove = (P(pk, cert_U), V(pk))$ protocol with a verifier V . Common input is pk and the private input to P is certificate $cert_U$. At the end of $Prove$ protocol V outputs either "accept" or "reject".*

Security properties of anonymous identification scheme is as follows:

- **Completeness:** $\Pr[Out_V = accept : P(pk, cert_U) \xleftrightarrow{Prove} V(pk) \rightarrow Out_V \wedge cert_U \leftarrow U(pk) \xleftrightarrow{Join} GM(pk, sk) \wedge (sk, pk) \leftarrow KeyGen(1^\lambda)] = 1$, when group manager GM , user U , prover P and verifier V are honest.
- **κ -Soundness:** $\Pr[Out_V = accept : P^* \xleftrightarrow{Prove} V(pk) \rightarrow Out_V] \leq \kappa$ in any of the following two cases; (i) $pk \notin L$, (ii) $pk \in L$ while algorithm P^* is independent from any $sk \in W'(pk)$.
- **Anonymous:** consider any PPT algorithm \tilde{V} , who will act as both GM and V in an attempt to break the anonymity of honest users. \tilde{V} runs $KeyGen$ and generates valid (sk, pk) . It then plays the following game: it interacts with a set of honest users, where

each user first `Join` and then multiple `Prove`. At some point \tilde{V} stops and outputs a bit. Let $p_{\text{real},\tilde{V}}(\lambda)$ be the probability that 1 is output.

Now we compare the above game to a different one, where \tilde{V} interacts with a simulator \mathcal{S} . The simulator gets as input the ordering and timing of `Join` and `Prove` protocols in the first game, and simulates the execution of those protocols in the same order and timing on behalf of users with random selection of inputs for `Prove` protocols from the generated certificates of `Join` protocols. Let $p_{\text{sim},\tilde{V}}(\lambda)$ be the probability that 1 is the output of \tilde{V} in this game.

We demand that there exists a PPT simulator \mathcal{S} such that for any \tilde{V} , we have $|p_{\text{sim},\tilde{V}}(\lambda) - p_{\text{real},\tilde{V}}(\lambda)| \leq \text{negl}(\lambda)$.

This definition of anonymity hides the identity of user even from the group manager. However, there are less demanding definitions that only require anonymity from the verifier [BMW03, NFHF09].

2.5 Proof Systems

The Σ -Protocol framework is also used for defining other cryptographic systems such as proof-of-knowledge schemes [Dam02, Gen04, KH06, GQ88, Sch91]. The proof-of-knowledge schemes are special proof systems that will be discussed in the following.

Definition 2.5.1 (Interactive Proof System [GMR89]) Let L be a language over $\{0,1\}^*$. Let (P,V) be an interactive protocol. We say that (P,V) is an interactive proof system for L if it satisfies the following two conditions:

- **Completeness:** If $x \in L$, then the probability that (P,V) rejects x is negligible in the length of x .
- **Soundness:** If $x \notin L$ then for any prover P^* , the probability that (P^*,V) accepts x is

negligible in the length of x .

The non-interactive version of a proof system is defined as follows: A non-interactive proof system for a language L allows one party P to prove membership in L to another party V for any $x \in L$. P and V initially share a string R of length polynomial in the security parameter λ . To prove membership of a string x in $L_\lambda = L \cap \{0, 1\}^\lambda$, P sends a message π as a proof of membership. Then V decides if to accept or to reject the proof. The shared string R is generated according to some distribution $U(\lambda)$ that can be generated by a probabilistic polynomial time Turing machine.

Let L be in NP. For any $x \in L$ there is a set of witnesses for its membership. Let $WL(x) := \{z \mid z \text{ is a witness for } x\}$. P is given a witness $z \in WL(x)$, but it is not available to V . Let $P(x, z, R)$ be the distribution of the proofs that P generates on input x , witness z and shared string R . Suppose that P sends V a proof π when the shared random string is R . Then the pair (R, π) is called the conversation.

Definition 2.5.2 (Non-Interactive Proof System [NY90]) *Let L be a NP language. We say that a pair of probabilistic Turing machines (P, V) , over distribution $U(\lambda)$, is a non-interactive proof system for L , if it satisfies the following two conditions:*

- **Completeness:** *(if $x \in L$ then P generates a proof that V accepts). For all $x \in L_\lambda$, for all $z \in WL(x)$, with overwhelming probability for $R \in_R U(\lambda)$ and $\pi \in_R P(x, z, R)$, V accepts on input (R, x, π) .*
- **Soundness:** *(if $x \notin L$ then no prover can generate a proof that V accepts). For all $y \notin L_\lambda$ with overwhelming probability over $R \in_R U(\lambda)$ and for all π , V rejects on input (R, x, π) .*

As we mentioned earlier, there is a variant of proof system (for both interactive and non-interactive), known as "proof-of-knowledge", where the prover's claim is a bit different: the prover claims to know a certain piece of information (such as a secret key corresponding to a given public one).

Definition 2.5.3 (Proof-of-Knowledge System [BG92]) *Let x be an element of language L in NP,*

and $W(x)$ the set of witnesses for x that should be accepted in the proof. Let R be a relation as $R = \{(x, w) : x \in L, w \in W(x)\}$, and κ be error function as $\kappa : \{0, 1\}^* \rightarrow \{0, 1\}$. The knowledge error $\kappa(x)$ denotes the probability that the verifier V accepts x , even though the prover does not know a witness $w \in W(x)$. A pair of interactive probabilistic Turing machines $(P(w, x), V(x))$, denoted as $\text{PoK}[w : x]$, are proof of knowledge for the relation R with knowledge error κ if the following two conditions hold:

- **Completeness:** $\Pr(P(x, w) \leftrightarrow V(x) \rightarrow 1) = 1$, if $(x, w) \in R$.
- **Soundness:** (no prover that does not know the witness can succeed in convincing the verifier). The protocol is κ -sound if there exists a polynomial-time machine E , given oracle access to \tilde{P} , such that for every \tilde{P} , it is the case that $\Pr[E^{\tilde{P}(x)}(x) \in W(x)] \geq \Pr[\tilde{P}(x) \leftrightarrow V(x) \rightarrow 1] - \kappa(x)$.

A general property that is applicable to proof protocols, including identification, authentication and proof-of-knowledge schemes, is zero-knowledge property. This property guarantees that the verifier does not learn anything about the secret of the prover. A few interpretations of zero-knowledge have been commonly considered in the literature [GMR89, For87], but here we describe only the original interpretation;

Definition 2.5.4 (Zero-Knowledge Protocol [GMR89]) A pair of interactive probabilistic Turing machines $(P(\alpha), V(z))$ is ζ -zero-knowledge for $P(\alpha)$, if for any PPT interactive machine $V^*(z, aux)$ there is a PPT simulator $S(z, aux)$ such that for any PPT distinguisher, any $(\alpha : z) \in L$, and any $aux \in \{0, 1\}^*$, the distinguishing advantage between the final view of V^* , in the interaction $P(\alpha) \leftrightarrow V^*(z, aux)$, and output of the simulator $S(z, aux)$ is bounded by ζ .

The key generation function (KeyGen) in Definition 2.4.1, that defines the relation between prover's private input and the verifier's input. An important property of this function is being one-way (Definition 2.5.5). This protects the secret key of the prover from other parties, including the verifier. The key generation function is produced by some cryptographic elements, such as commitment

and encryption schemes. Here we define these elements;

Definition 2.5.5 (One-way Function [Gol01]) *By considering λ as the security parameter, an efficiently computable function $OUT \leftarrow f(IN)$, is one-way if there is no PPT algorithm that takes OUT as input and returns IN with non-negligible probability in terms of λ .*

2.6 Commitment

Another cryptographic primitive is commitment scheme that allows one to commit to a chosen statement while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes are designed so that a party cannot change the statement after they have committed to it. In the following, we formally define them.

Definition 2.6.1 (Commitment Scheme [Gol01]) *A (non-interactive) Commitment Scheme (for a message space \mathbb{M}) is a triple $(\text{Setup}, \text{Commit}, \text{Open})$ such that:*

- (a) $CK \leftarrow \text{Setup}(1^\lambda)$ generates the public commitment key.
- (b) for any $m \in \mathbb{M}$, $(c, d) \leftarrow \text{Commit}_{CK}(m)$ is the commitment/opening pair for m . $c = c(m)$ serves as the commitment value, and $d = d(m)$ as the opening value. We omit mentioning the public key CK when it is clear from the context.
- (c) $\text{Open}_{CK}(c, d) \rightarrow \tilde{m} \in M \cup \{\perp\}$, where \perp is returned if c is not a valid commitment to any message.

The security properties of commitment schemes are threefold: (1) honest opening of and honest commitment of any m returns the same m , (2) c gives no information about m , and (3) It is not possible to open c in two different ways. The properties stated above are correspondingly called correctness, hiding and binding.

- **Correctness:** for any $m \in M$ and $CK \leftarrow \text{Setup}(1^\lambda)$, $\text{Open}_{CK}(\text{Commit}_{CK}(m)) = m$

- **Hiding:** *it is computationally hard for any adversary \mathcal{A} to generate two messages $m_0, m_1 \in M$ such that adv can distinguish between their corresponding commitments c_0, c_1 . That is, $c(m)$ reveals no information about m . In other words, for any PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we require:*

$$\Pr \left[b = \tilde{b} \mid \begin{array}{l} CK \leftarrow \text{Setup}(1^\lambda), (m_0, m_1, \alpha) \leftarrow \mathcal{A}_1(CK), \\ b \leftarrow_R \{0, 1\}, (c, d) \leftarrow \text{Commit}(m_b), \tilde{b} \leftarrow \mathcal{A}_2(c, \alpha) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

- **Binding:** *it is computationally hard for the adversary \mathcal{A} to come up with a triple (c, d, d') , referred to as a collision, such that (c, d) and (c, d') are valid commitments for m and m' when $m \neq m'$. In other word, for any PPT \mathcal{A} we require:*

$$\Pr \left[\begin{array}{l} m \neq m' \wedge \\ m, m' \neq \perp \end{array} \mid \begin{array}{l} CK \leftarrow \text{Setup}(1^\lambda), (c, d, d') \leftarrow \mathcal{A}(CK), \\ m \leftarrow \text{Open}(c, d), m' \leftarrow \text{Open}(c, d') \end{array} \right] \leq \text{negl}(\lambda)$$

The hiding property of commitment schemes is similar CPA-security (Definition 2.1.2) of public-key encryption schemes. For this reason, some identification scheme use public-key encryption schemes. On the other hand, the binding property of commitment schemes is similar to security of signature scheme (Definition 2.2.1). In commitment schemes, since $c = c(m)$ cannot be opened to any other value than m , then it implies that in a sense c validates m .

Algorithm 2.6.1 (Pedersen Commitment [Ped92]) $\text{Commit}(x, r) = g^x h^r \bmod p$, where p is prime chosen with λ bit security, g and h are prime group generator for \mathbb{Z}_p , and $x, r \in \mathbb{Z}_p^*$.

A special case of commitment scheme is bit commitment that is defined as follows:

Definition 2.6.2 (Bit Commitment [Nao91]) A commitment scheme (Definition 2.6.1), is a bit commitment when the message space is $\mathbb{M} = \{0, 1\}$.

A general algebraic property is homomorphism that is used in many cryptographic primitives, such as commitment, encryption and signature. **Homomorphism** is a structure-preserving map between two algebraic structures of the same type.

Property 2.6.1 (Homomorphism [Bir40]) Consider a map $f : \mathbb{A} \rightarrow \mathbb{B}$ between two sets \mathbb{A}, \mathbb{B}

equipped with the same structure such that, if $*$ is an operation of the structure. f is homomorphic if $f(x * y) = f(x) * f(y)$ for every pair (x, y) of elements of \mathbb{A} .

Here, we consider homomorphism of bit commitment schemes in a specific way;

Definition 2.6.3 (Homomorphic Bit Commitment [Gro03]) Consider a bit commitment scheme (Definition 2.6.2), where algorithm Commit is taking $b \in \mathbb{Z}_2$ and $\rho \in \mathbb{G}$ as input, and returns $\text{Commit}(b, \rho) \in \mathbb{G}$. This scheme is homomorphic if the following holds: $\forall b, b' \in \mathbb{Z}_2$ and $\forall \rho, \rho' \in \mathbb{G}$, we have $\text{Commit}(b, \rho) \text{Commit}(b', \rho') = \text{Commit}(b + b', \rho\rho')$.

Homomorphic bit commitment can be obtained using Goldwasser-Micali encryption, as follows;

Algorithm 2.6.2 (Goldwasser-Micali Bit Commitment [GM84]) $\text{Commit}(b, \rho) = \theta^b \rho^2 \pmod{N}$, where $N = pq$ for secret prime values of p and q chosen with λ bit security, θ is a quadratic residue modulo N , $b \in \{0, 1\}$ and $\rho \in \mathbb{Z}_N^*$.

Camenisch and Lysyanskaya [CL04] proposed a special signature scheme (Definition 2.2.1), that allows the users to commit to a message and then prove the knowledge of a signature on the message without leaking information about the message. This scheme is based on the standard definition of signature schemes due to Goldwasser, Micali, and Rivest [GMR88].

Definition 2.6.4 (CLSig [CL04]) CLSig is a tuple $(T, PK, SK, S, V, \text{KeyGen}, \text{Sign}, \text{Verify}, \text{BSign}, \text{SPK})$, where the first eight parameters are as defined in (Definition 2.2.1), and the additional last two parameters are as follows;

- (V) BSign is a blind signature protocol between a prover P , who takes message $M \in \{0, 1\}^*$ and public key pk as input, and the signature authority A , who takes private key sk as input. The prover first commits to the message M , and then interact with the signature authority to generate a valid signature σ on the message M , that satisfies $\text{accept} \leftarrow \text{Verify}_{pk}(M, \sigma)$. At the end of protocol, both prover and signature authority output a valid signature σ on the message M , while the signature authority does not learn any information about M .

(VI) SPK is a protocol between a prover P , who takes message $M \in \{0,1\}^*$ and signature σ as input, and a verifier V , who takes public key pk as input. The prover first commits to a message M and then convinces the verifier that is in possession of the message M and a signature σ that satisfies $\text{accept} \leftarrow \text{Verify}_{pk}(M, \sigma)$, without leaking information about M or σ . At the end of protocol, V outputs the commitment of message M and $\text{Out}_V = 1$ if they accept, or $\text{Out}_V = 0$ if they reject, while P outputs the randomness of the commitment.

Security properties of CLSig are:

- **Correctness:** $\Pr[\text{Out}_V = \text{accept} : P(M, \sigma) \xleftrightarrow{\text{SPK}} V(pk) \rightarrow \text{Out}_V \wedge \sigma \leftarrow P(M, pk) \xleftrightarrow{\text{BSign}} SA(sk) \wedge (sk, pk) \leftarrow \text{KeyGen}(1^\lambda)] = 1$, when signature authority SA , prover P and verifier V are honest.
- **Unforgeability:** Same as *security* property of signature schemes, as in Definition 2.2.1.
- **κ -Sound:** $\Pr[\text{Out}_V = \text{accept} : P^* \xleftrightarrow{\text{SPK}} V(pk) \rightarrow \text{Out}_V] \leq \kappa$ if P^* is independent from any (M, σ) that $\text{accept} \leftarrow \text{Verify}_{pk}(M, \sigma)$.
- **Zero-Knowledge;** both BSign and SPK are *zero-knowledge* protocols according to Definition 2.5.4.

BBS⁺ [CL04] is an example of CLSig scheme.

2.7 Theorems

Here we describe some mathematical theorems that will be used in security proofs.

Theorem 1 (Chernoff-Hoeffding Bound [Che52], [Hoe63]) For any values (ϵ, n, τ, q) where $\epsilon > 0$, $n \geq \tau \geq 0$ and $1 \geq q \geq 0$, we have the following inequalities about the following function

$$\text{Tail}(n, \tau, q) := \sum_{i=\tau}^n \binom{n}{i} q^i (1-q)^{n-i};$$

- if $\frac{\tau}{n} < q - \epsilon$, then $\text{Tail}(n, \tau, q) > 1 - e^{-2\epsilon^2 n}$

- if $\frac{\tau}{n} > q + \varepsilon$, then $Tail(n, \tau, q) < e^{-2\varepsilon^2 n}$

2.8 Concluding Remarks

This chapter presented background concepts and definitions used throughout the thesis. We reviewed some basic definitions of cryptographic primitives such as proof systems, proof-of-knowledge, authentication, identification, commitment, encryption, and signature schemes.

Chapter 3

Literature Review

In this chapter we describe two major class of authentication schemes; challenge-response-based and timestamp-based. We discuss the major works of the distance-bounding literature as a special class of challenge-response authentication schemes and then we discuss some of the existing models that consider the concept of time in entity authentication.

Entity authentication is the process by which an agent in a distributed system gains confidence in the identity of a communication partner. Authentication is one the most important security goal in cryptographic protocols. In authentication schemes, time is used for many different purposes: have expiry time for public keys and certificates, to reduce their breaking impact to guarantee the message freshness. For each of these purposes, a special model is necessary to formally define the security goals. Replay attacks are considered as major threat against authentication protocols, so the message freshness need be guaranteed. In practice, this is achieved by including fresh values into messages. These fresh values are in the form of nonces or challenges chosen by the other party, or timestamps.

- **Authentication with nonces.** Nonces are given through messages to a party p , and p replies with a message that includes the nonce value. This can be modeled by input tape of a Turing machine. In one-sided authentication, this implies at least two messages protocol, and for mutual authentication at least three messages protocol [Sch14].
- **Authentication with challenge-response.** Challenges are considered as random

questions that are given through messages to a party p , and p replies with a message that are considered as the answer. A correct answer guarantees the freshness of response. This can be modeled by input tape of a Turing machine.

- **Authentication with timestamps.** Timestamps are more difficult to model, because a Turing machine cannot model real time [Sch14, Section 1]. However, timestamps enable us to design more efficient protocols, with less latency. Thus timestamps can be seen as a replacement for challenges, but they are not identical: they allow us to design more efficient protocols in terms of number of messages, and provide some ordering. On the downside, the receiver must keep an ephemeral local state to detect the replay of valid messages within an acceptance window.

In the section (Section 3.1) we provide a brief review about different aspects of distance-bounding literature, that can be classified as challenge-response-based authentication. And in the following section (Section 3.2) we discuss some of the important models that consider time in entity authentication.

3.1 Distance-Bounding as Challenge-Response Entity Authentication

In this section we introduce the fundamentals of distance-bounding, as a special class of entity authentication. Then we describe three classes of distance-bounding protocols based on the input setting of participants: symmetric key, public key and anonymous DB. All the existing distance-bounding schemes, including the schemes that will be presented in this section, fit in "authentication with challenge-response" class. Later in this thesis, we propose the first distance-bounding scheme that fits in "authentication with timestamps" class.

Distance-Bounding schemes have been introduced to defeat relay attack in authentication protocols, where a MiM attacker only relays the messages between prover and verifier [BC94]. A DB protocol relies on the fact that no message can travel faster than light. So the verifier can find

the distance of a prover to the verifier by measuring the round-trip time of a challenge and the corresponding response. The main goal is that a *prover*, holding secret key, prove an upper bound on their distance to a *verifier*. A general sketch of a typical DB protocol is shown in Figure 1.3 between a prover P and a verifier V .

Ideally, a distance-bounding scheme should have the properties of authentication schemes:

- *Completeness*: an honest prover who is located close to the verifier, will succeed in the protocol with high probability.
- *Soundness*: if the verifier accepts in a protocol, then the information held by the close-by participants to the verifier, includes the secret key.
- *Security*: if the prover behaves honestly in a protocol, then the provided information by them, does not provide any advantage to defeat *soundness*.

There are three important attacking scenarios in the distance-bounding systems that are new compared to the attack scenarios of authentication schemes. These three attacks are the main focus of protocol designers in this field:

- **Attack 3.1.1** *Distance-Fraud Attack (DF)[BC94]*. A dishonest prover \mathcal{P}^* , which is not located within distance bound \mathcal{B} to verifier \mathcal{V} , tries to convince \mathcal{V} that she is authentic and located within the distance bound \mathcal{B} to \mathcal{V} (See Figure 3.1).

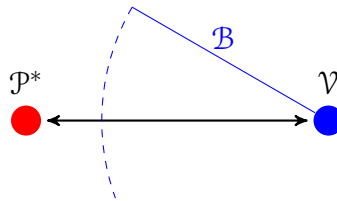


Figure 3.1: Distance Fraud Attack

- **Attack 3.1.2** *Mafia-Fraud Attack (MF)[Des88]*. This is a Man-In-the-Middle attack between honest prover \mathcal{P} (far away) and verifier \mathcal{V} . Adversary \mathcal{A} , which is

located within the distance bound \mathcal{B} to \mathcal{V} , commits the fraud. This attack convinces \mathcal{V} that \mathcal{P} is located within the distance bound \mathcal{B} to \mathcal{V} (See Figure 3.2).

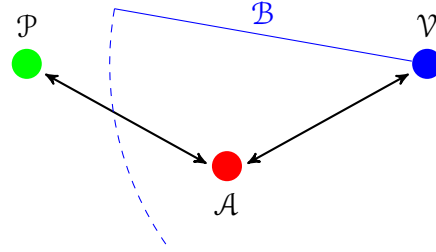


Figure 3.2: Mafia Fraud Attack

- **Attack 3.1.3 Terrorist-Fraud Attack (TF)**[Des88]. This is a co-operative fraud running by a far-away dishonest prover \mathcal{P}^* and a helper \mathcal{H} that is located within the distance bound \mathcal{B} to \mathcal{V} . The attack happens with minimal disclosure about secret-key of the prover to the helper. \mathcal{P}^* and \mathcal{A} co-operate to convince \mathcal{V} that \mathcal{P}^* is located within distance bound \mathcal{B} to \mathcal{V} (See Figure 3.3).

In the *original TF-resistance*, it's assumed that the prover does not leak their secret key to the actor. In the *recent TF-resistance* definition [Vau13] this restriction is removed, but it is required that non-negligible success of TF attack results in non-negligible improvement in future impersonation attacks by the actor.

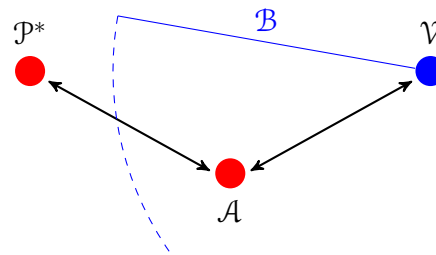


Figure 3.3: Terrorist Fraud Attack

There are two other minor attacking scenarios in this field that are considered as special case of the above main attacks in some works:

- **Attack 3.1.4** *Distance-Hijacking*[CRSC12]. In this attack a dishonest prover \mathcal{P}^* , which is not located within the distance \mathcal{B} to a verifier \mathcal{V} , exploits some honest provers $\mathcal{P}_1, \dots, \mathcal{P}_n$ with different secret-keys to mislead \mathcal{V} about the actual distance between \mathcal{P}^* and \mathcal{V} (See Figure 3.4). This attack is considered as a special DF attack in some models [VBM⁺13].

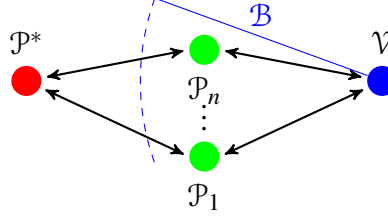


Figure 3.4: Distance Hijacking Attack

- **Attack 3.1.5** *Impersonation*[ABK⁺11]. In this attack a close-by dishonest prover \mathcal{P}^* purports to be another prover \mathcal{P} in her interaction with \mathcal{V} (See Figure 3.5). This attack is considered as a special MF attack in some models [VBM⁺13].

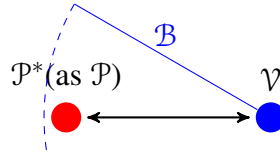


Figure 3.5: Impersonation Attack

3.1.1 Symmetric Key Distance-Bounding

In a symmetric key distance-bounding protocol, a prover and a verifier share a secret key value and then they run the distance-bounding protocol. Although the seminal distance-bounding paper Brands-Chaum [BC94] does not have symmetric settings, but majority of protocols in this field are symmetric. The generic symmetric DB protocol is shown in Figure 3.6. This is an example of the general protocol in Figure 1.3 where both parties take the same secret input. Because the

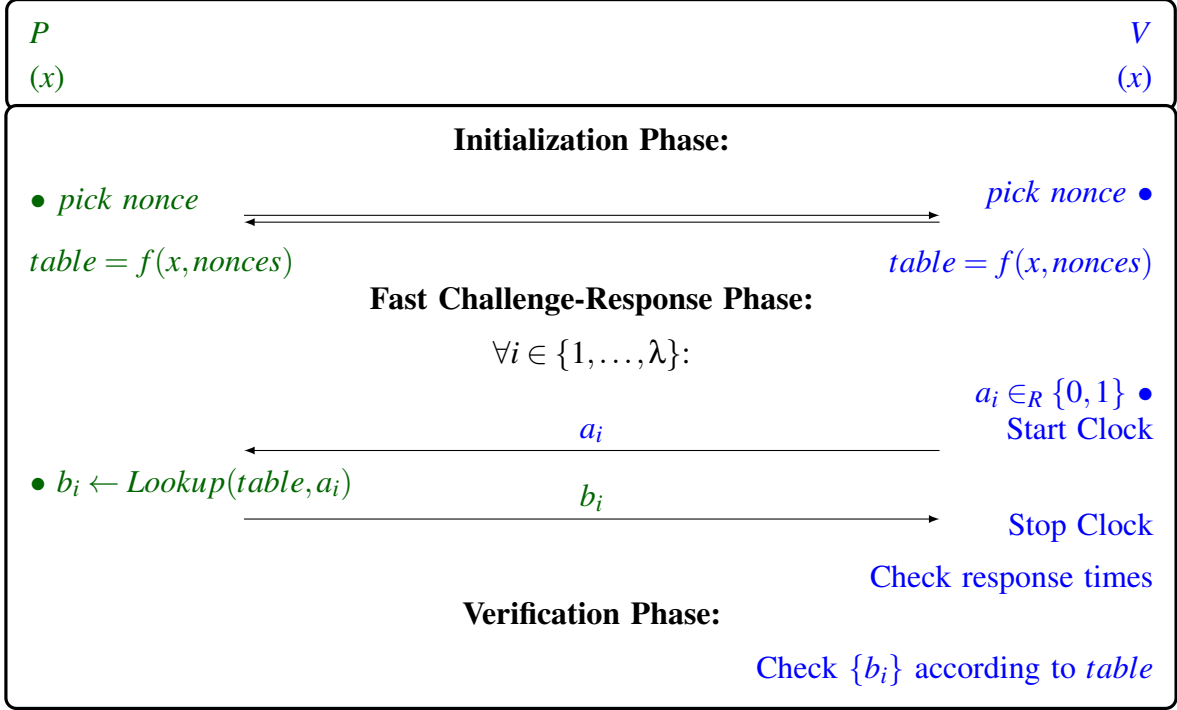


Figure 3.6: General sketch of symmetric key distance-bounding protocols. x is the shared key with length λ . In initialization phase, the parties exchange randomness, that allows them to generate a table. In fast phase, the verifier asks about half values of the table and measures response time. In verification phase, the verifier checks validity of responses in respect to the table.

focus of our work is on public-key DB, in this section we talk about only one symmetric distance-bounding protocol, and leave the more comprehensive review of this literature in Appendix A.1. Swiss-Knife [KAK⁺08] is one of the strongest symmetric key DB protocols that is secure against all distance-bounding protocols.

Kim *et al.* [KAK⁺08] designed one of the first DB protocols that is secure in all three of DB security models (i.e. DF , MF and TF). The presented protocol is named Swiss-Knife, which is based on Map1 [BR93] mutual-authentication protocol, and inherits the mutual authentication property, in which, both involved parties of protocol prove their authenticity to the other party. Moreover, this protocol supports noisy channels and privacy of provers against any eavesdropper.

Kim *et al.* proposed a new MiM attack, which makes it hard to provide TF resistance protocol

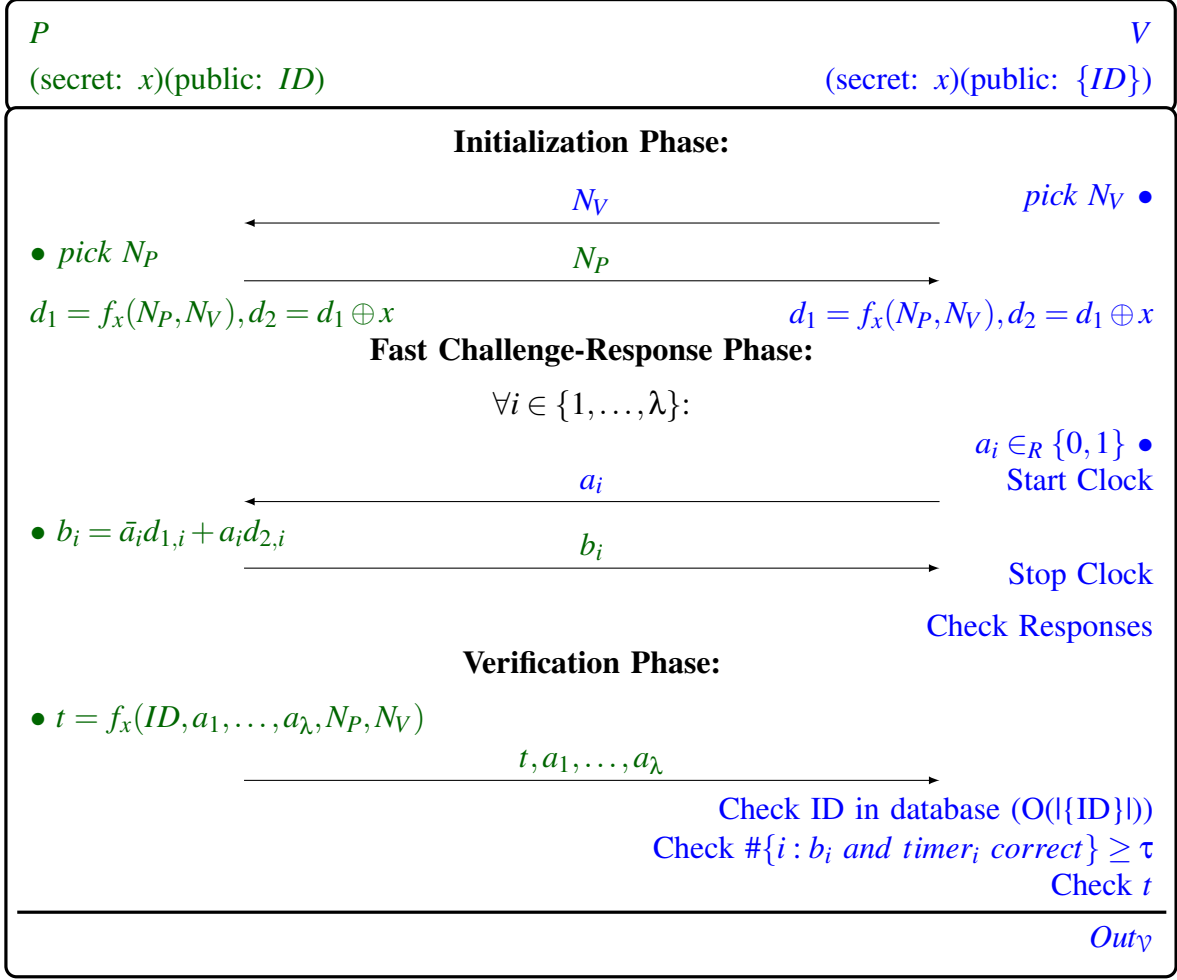


Figure 3.7: Swiss-Knife distance-bounding protocol [KAK⁺08]. x is the shared key with length λ , ID is identity of prover and $\{ID\}$ is identity list of all provers. $f_x()$ is a pseudo random function.

without leaking information about the secret-key. This attack allows an adversary to recover the long-term key x . In order to learn bit x_i , the adversary can, during the fast bit exchange, flip the value of challenge bit a_i when it is transmitted from verifier to prover and leave all other messages untouched. The adversary then observes the verifier's reaction; if the verifier accepts, it means that the prover's answer b_i was nevertheless correct, which may lead to leak the value of x_i , as is does in Tu-Piramuthu protocol [TP07].

3.1.2 Public-Key Distance-Bounding

In a public-key distance-bounding protocol, provers have a key pair and the verifier has access to the public-key of provers before running the distance-bounding protocol. The direct way of constructing a public-key DB protocol, is to first establish a shared session key between the prover and the verifier by using the public-key setting, and then run a symmetric-key DB protocol. However, by using the direct method we cannot achieve the TF resistance property, as the malicious prover can run the key establishment phase and pass the session key to the helper.

The generic public-key DB protocol is shown in Figure 1.3. In this section we review some of the important public-key distance-bounding protocols.

Brands-Chaum [BC94]

Brands-Chaum designed the first DB protocol in order to prevent *relay* attacks. In this model, each prover has a key pair. The verifiers are trusted and have access to the public key of provers. The proposed protocol (Figure 3.8) is designed to be secure against DF and MF attacks. This protocol does not tolerate the presence of environment noise.

Bussard-Bagga [BB04] (DBPK-Log)

Bussard-Bagga proposed the first public-key DB protocol to be secure against TF attack.

This protocol combines a *fast-exchange* DB protocol, Pedersen commitment scheme [Ped92] and *zero-knowledge proof-of-knowledge* [PHS03]. Before running the protocol, a prover chooses a key pair and registers the public-key with a trusted authority. The verifiers are trusted and have access to the public-key of provers. The system parameters are set by a trusted authority, which are shared with all participants.

The main difference between this protocol compared to Brands-Chaum [BC94] is that the fast challenge-response table is not completely random. In fact knowing the table allows to calculate

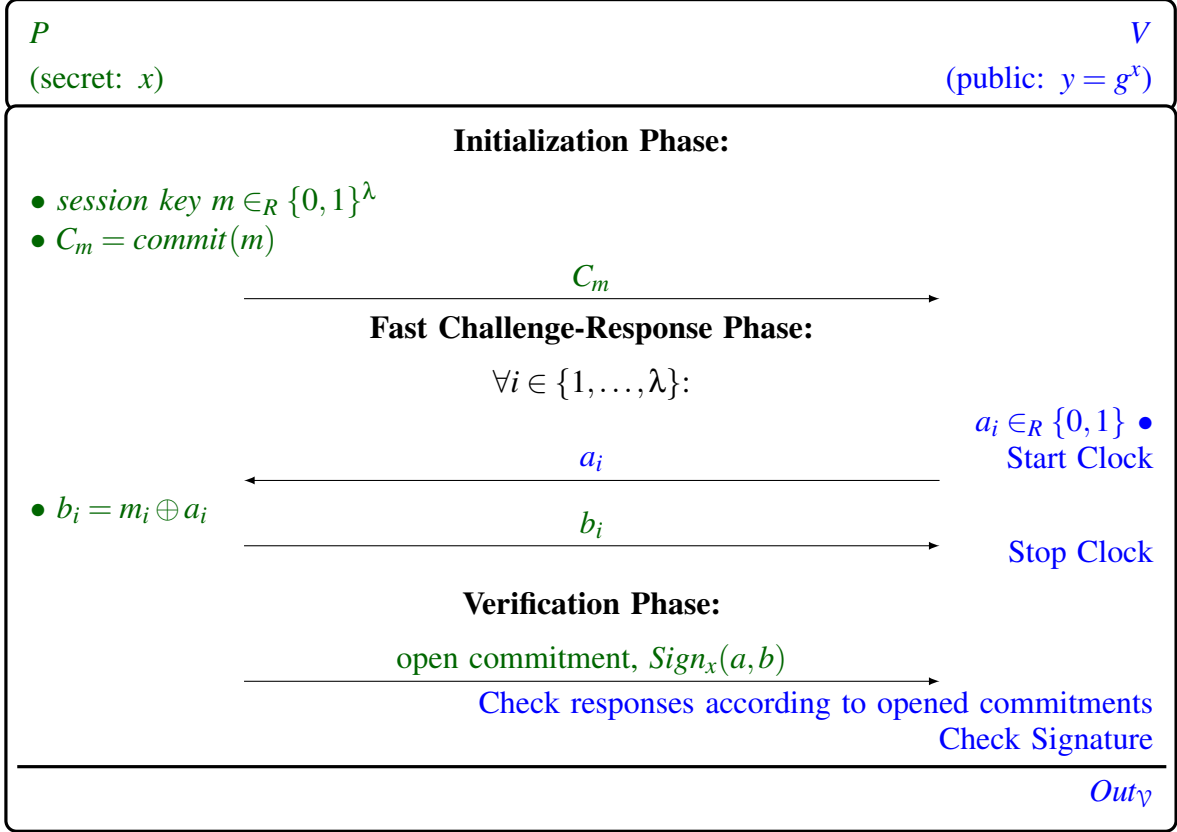


Figure 3.8: Brands-Chaum Protocol [BC94]. x is prover's private key with length λ and y is prover's public key. Prover commits to a random table and then runs the fast phase. In verification phase, prover opens the committed table.

the private-key of prover. This technique is used to makes the protocol resistant against TF attack.

The DB protocol combines the bitwise operations of the fast *challenge-response* phase, and mathematical operations that are used in commitment schemes. This results in some security loss of the secret-keys, while maintaining indistinguishability of the secret-keys. Figure 3.9 is showing the construction of this protocol. Note that this protocol does not tolerate noisy channels.

In commitment opening phase, only half of commitments are opened and the other half are unused. Bay *et al.* [BBM⁺13] proposed *TF* and *DF* attacks on this protocol, which takes advantage of poor auditing of un-used elements in the Commitment Opening phase, in which half of the bit commitments would not be opened. See Bay *et al.* [BBM⁺13] for the details of this attack.

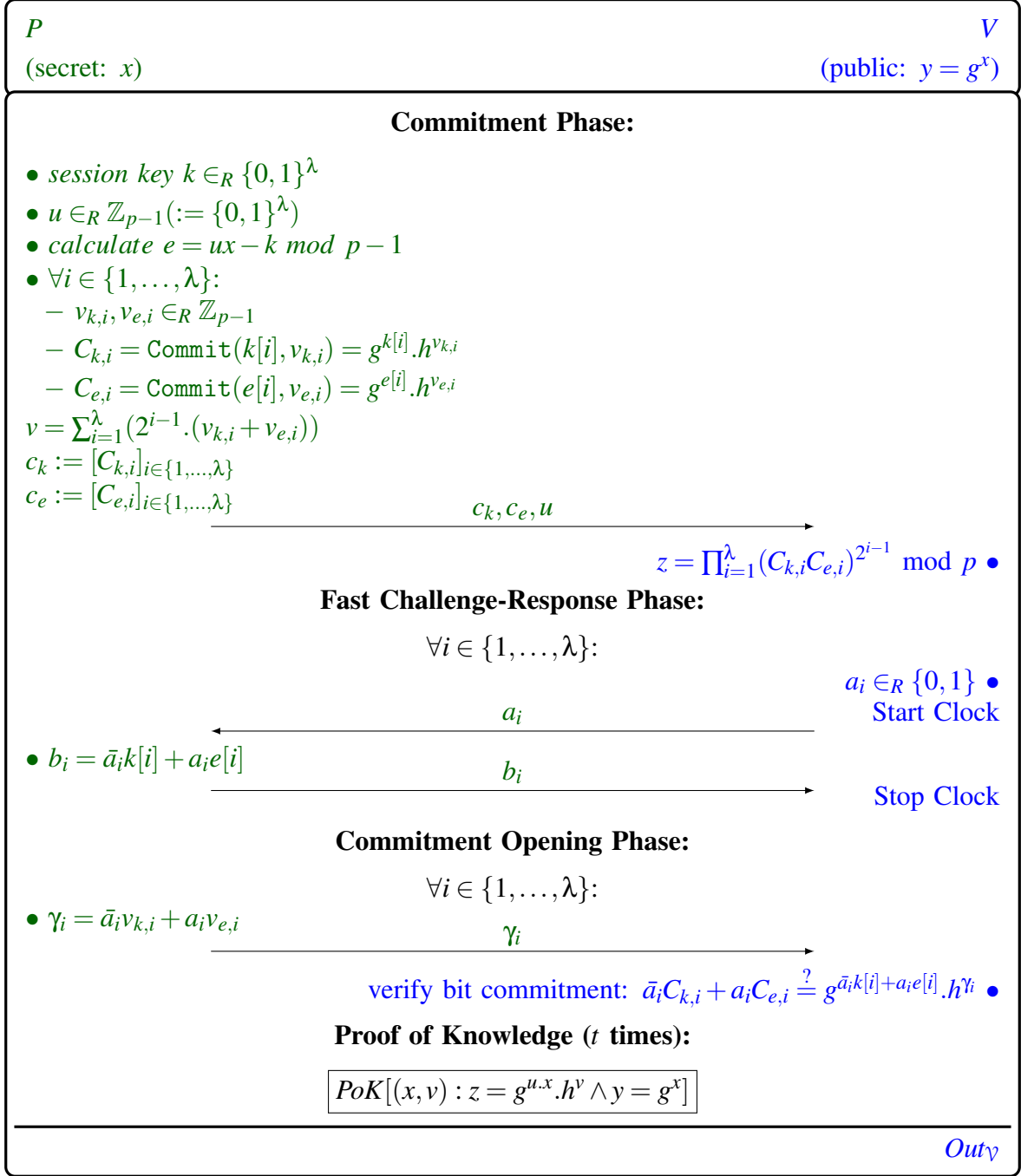


Figure 3.9: DBPK-log Protocol. x is prover's private key with length λ and y is prover's public key. t and λ are security parameters.

Hermans *et al.* [HPO13]

Hermans *et al.* designed a public-key DB protocol that is secure against *DF* and *MF* attacks. However, it is vulnerable to *TF* attack, since it's using the trivial construction method that is stated at

the beginning of this section.

The proposed protocol (Figure 3.10) has a public-key setting, which arises from the desire to design a protocol without key updates, that guarantees stronger privacy. In this protocol all provers are initialized with a private/public key pair $(x, X = xP)$ and the provers' public keys are registered in the verifier's database. The private/public key pair of the verifier is $(y, Y = yG)$ of which the public-key is known to all provers.

To generate the ephemeral shared secret, an anonymous Diffie-Hellman key agreement, with fresh random values from both sides ($R_1 = r_1G$ and $R_3 = r_3G$), takes place, resulting in a shared point r_1r_3G on the elliptic curve.

Gambs *et al.* [GKL⁺14]

Gambs *et al.* [GKL⁺14] proposed a public-key DB protocol (called VSSDB) similar to Hermans *et al.* [HPO13], that is claimed to be secure against DF, MF and TF. The presented protocol (Figure 3.11) is between prover and verifier, where prover has access to the public-key of verifier and its own secret key. The verifier has access to its private key and the public key of prover.

We later present a new TF attack against VSSDB protocol.

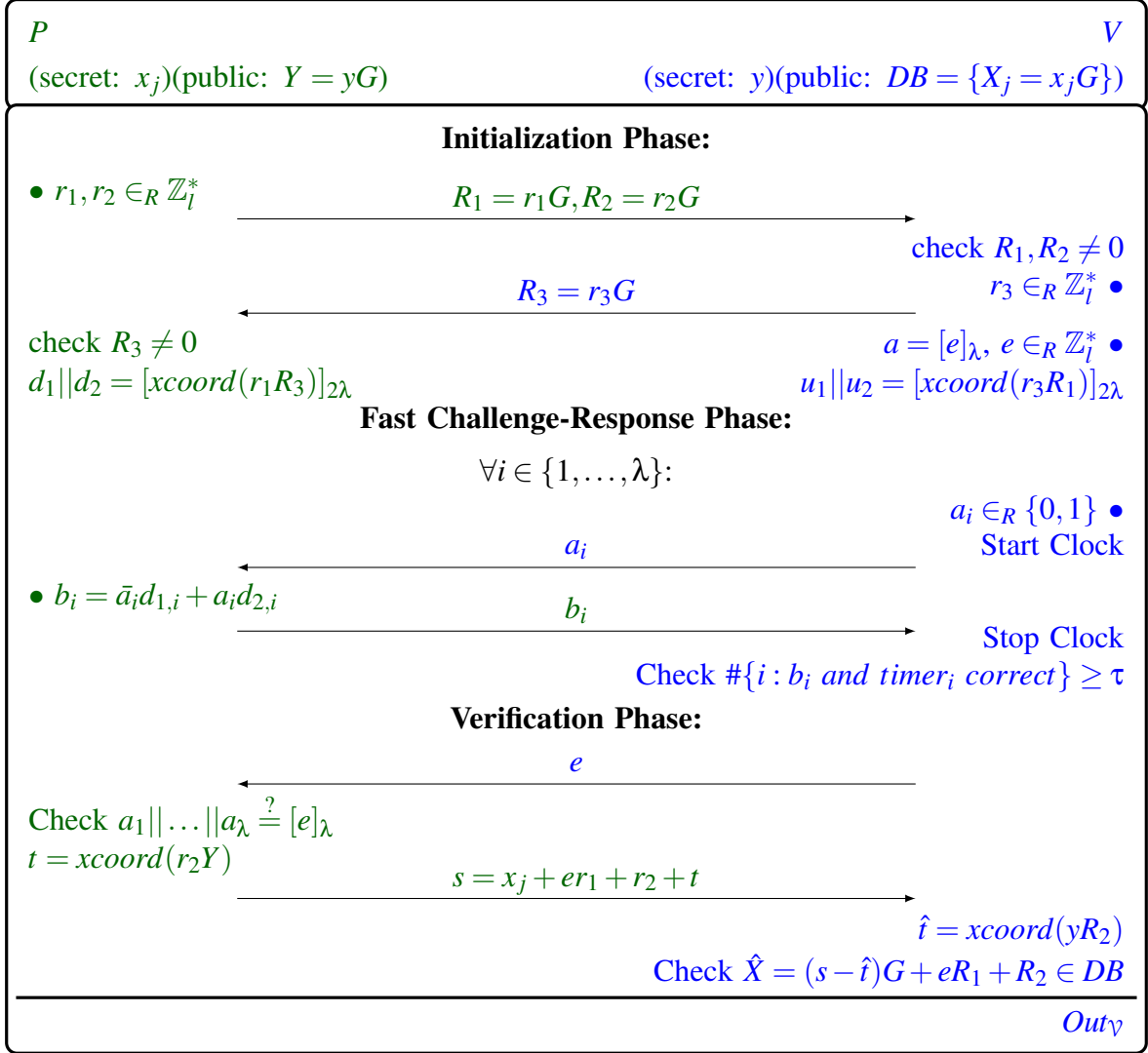


Figure 3.10: HPO Protocol. x_j is prover's private key with length λ and X_j is prover's public key. y is verifier's private key and Y is verifier's public key. $DB = \{X_j\}$ is public-key list of all provers. G is a generator point on elliptic curve and $xcoord(.)$ returns X coordinate of a point, which is statistically indistinguishable from the uniform distribution.

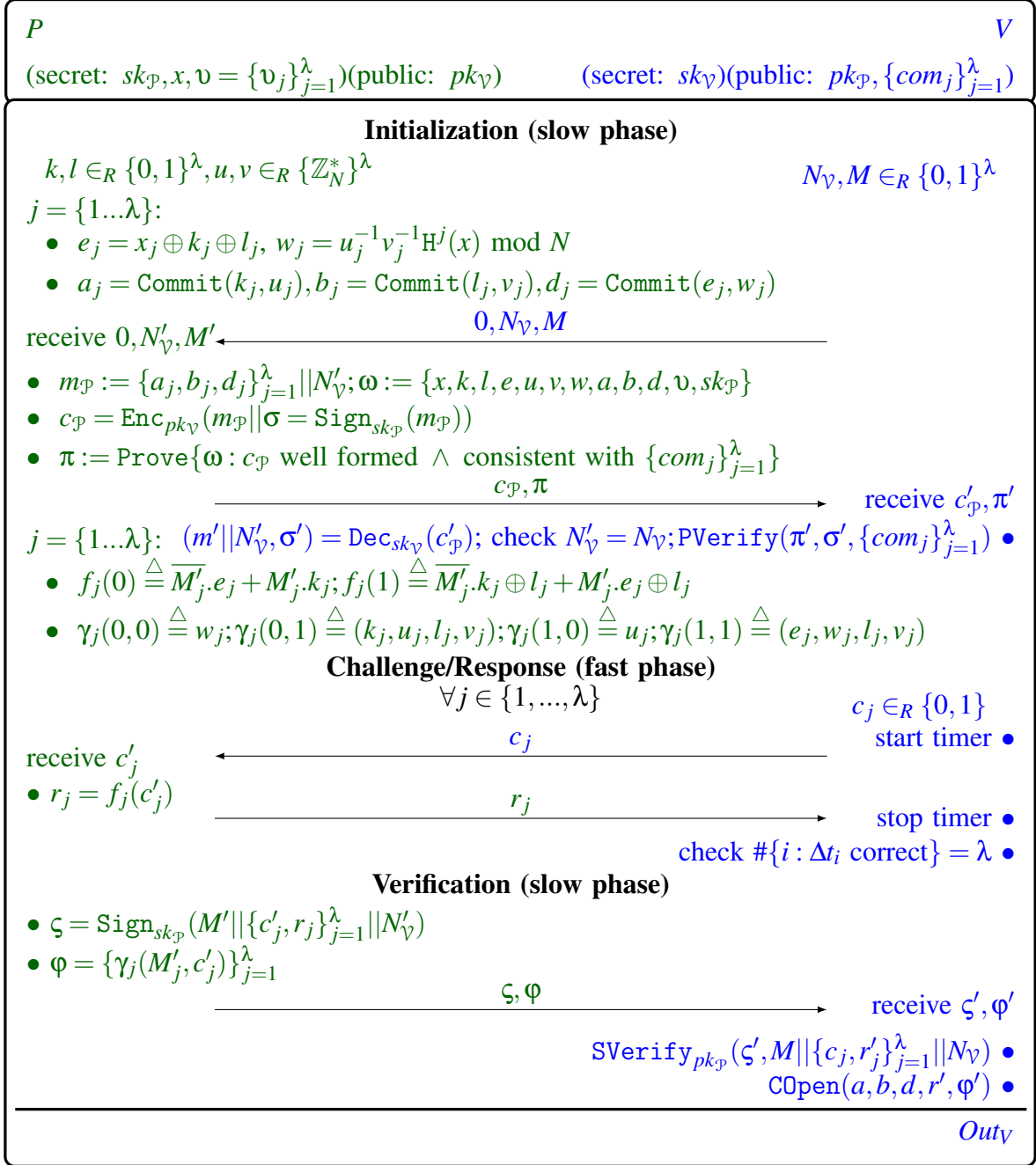


Figure 3.11: VSSDB public-key DB protocol. (sk_P, x) is prover's private key where x has random distribution with length λ and pk_P is prover's public key. sk_V is verifier's private key and pk_V is verifier's public key. $(\text{Commit}, \text{COpen})$ is a commitment scheme. (Enc, Dec) is a secure public key encryption scheme. $(\text{Sign}, \text{SVerify})$ is a signature scheme. $(\text{Prove}, \text{PVerify})$ is a proof-of-knowledge scheme. H is a secure hash function with pseudo-random output. $v_j = H^j(x)$ and $com_j = \text{Commit}(x_j, v_j)$ for $j = 1 \dots \lambda$.

Vaudenay [Vau14]

Vaudenay [Vau14] presents a formal model for public-key DB protocol that is an extension of proof-of-knowledge schemes. They also propose a public-key DB protocol (called ProProx) that uses Goldwasser-Micali encryption for public-key generation.

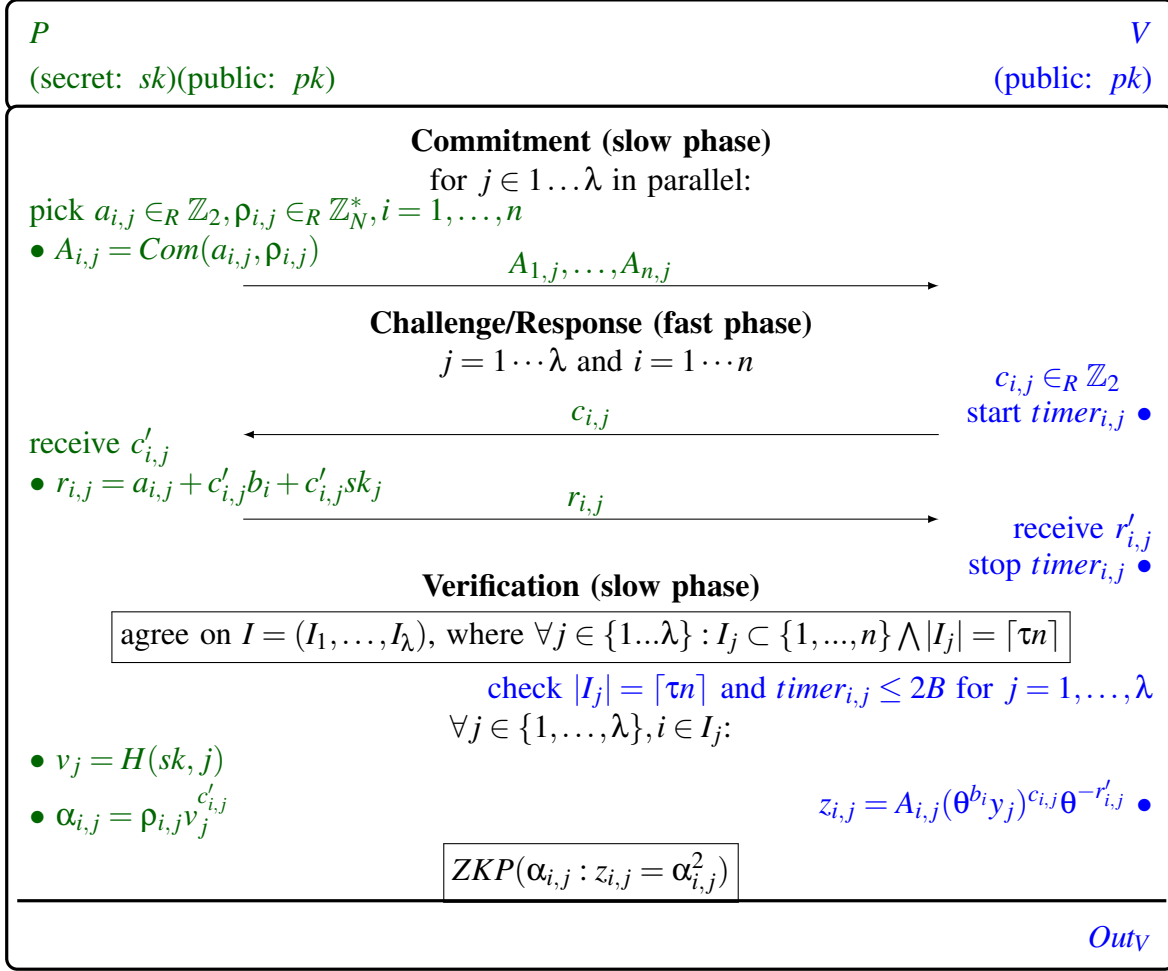


Figure 3.12: ProProx public-key DB protocol. sk is prover's private with length λ and pk is prover's public key. $Com(.,.)$ is Goldwasser-Micali encryption. τ is the minimum threshold ratio of noiseless fast rounds. ZKP is an interactive zero-knowledge proof. The number of fast rounds is $n\lambda$. In each fast round, the verifier sends one-bit challenge, and receives the corresponding response.

This protocol is presented in Figure 3.12. This protocol is proven to be secure against DF, MF and

TF attacks. An advantage of this work compared to the last two protocols, is that the verifier does not need to be registered, which expands the range of applications.

In the verification phase of Figure 3.12, the prover and the verifier agree on a list $I = (I_1, \dots, I_\lambda)$, where each I_j consists of $\lceil \tau n \rceil$ indices from 1 to n . Both parties believe $\forall j = \{1 \dots \lambda\}, i \in I_j : c_{i,j} = c'_{i,j}$ and $r_{i,j} = r'_{i,j}$. The verifier then checks whether responses are within the required time interval. The prover and the verifier then run an interactive zero-knowledge proof (ZKP) to show that the responses $r_{i,j}, j = \{1 \dots \lambda\}, i \in I_j$ are consistent with the corresponding $A_{i,j}$'s and y_j 's. If the verification fails, the verifier aborts and outputs $Out_v = 0$, otherwise, outputs $Out_v = 1$.

3.1.3 Anonymous Distance-Bounding

Distance-bounding protocols reveal the identity of the prover to the verifier: in symmetric key DB, the prover and the verifier share a secret key, and in public-key DB, the prover's response is compared against the public-key of a specific user. In anonymous DB however the goal is to prove that the distance of a registered user is less than a prescribed bound, without revealing their exact identity.

In an anonymous distance-bounding protocol, each registered prover has a certificate as proof of membership of a group. By having a certificate, the prover starts the DB protocol with a verifier who has only access to the public parameters of the system.

In this section we discuss all of the existing anonymous distance-bounding protocols. Some other privacy notions of prover is discussed in Appendix A.3.

Gambs *et al.* [GOR14]

GOR is an anonymous DB system [GOR14] that authenticates the provers as a member of an authorized group. The novelty of this work is in using the homomorphic EC+ElGamal encryption scheme. GOR uses accumulation technique to provide anonymity of provers.

GOR is built based on the structure of HP0 public-key DB protocol (Figure 3.10 [HPO13]), that does not provide TF-resistance. As a result, GOR is not TF-resistance either. Figure 3.13 presents the GOR protocol between the prover and the verifier, in which the prover has access to the public-key of the verifier and a public and private key pair that is used for proving group membership, and the verifier has access to its private-key and the public-key of all provers. The anonymity of this scheme is shown to be broken in [Vau16].

Bultel *et al.* [BGG⁺16]

SPADE [BGG⁺16] is an anonymous DB system that authenticates a prover as a member of an authorized group. This ensures anonymity, because authentication relies on a group signature. The verifier in SPADE must be registered and have a key-pair of its own. Figure 3.14 presents the SPADE protocol between the prover and the verifier. The prover has access to the public-key of the verifier and a secret-key that is used for generating the group signature, and the verifier has access to its private-key and the group public-key. This protocol is designed to be secure against DF, MF and TF attacks. However, we later propose a TF attack against it in Section 5.1.1.

Avoine *et al.* [ABG⁺17]

TREAD [ABG⁺17] is an anonymous DB system that authenticates the provers as a member of an authorized group. The protocol ensures anonymity because authentication uses a group signature. The verifier needs to be registered and have a key-pair of their own. The structure of TREAD is very similar to SPADE. Figure 3.15 presents TREAD protocol between the prover and the verifier, in which the prover has access to the public-key of the verifier, a secret-key that is used for generating group signatures and a pair of public and private identities, and the verifier has access to its private-key and the group public-key. Note that TREAD is designed in a way that can operate as public-key DB too. This protocol is designed to be secure against DF, MF and TF attacks.

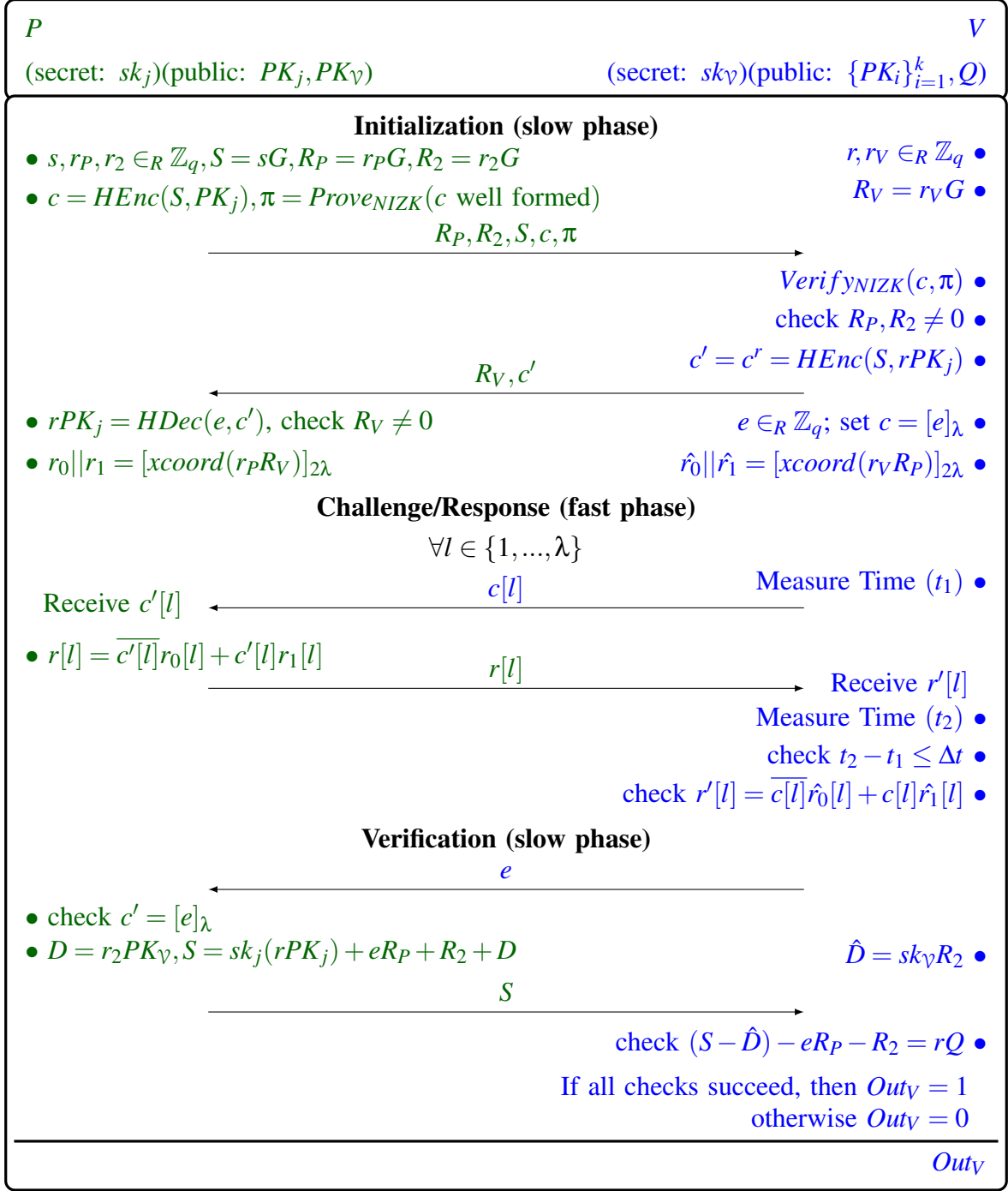


Figure 3.13: GOR anonymous DB protocol. λ is security parameter. sk_j is prover's private and PK_j is prover's public key. sk_V is verifier's private and PK_V is verifier's public key. $(HEnc, HDec)$ is a homomorphic encryption scheme. $(Prove_{NIZK}, Verify_{NIZK})$ is a non-interactive zero-knowledge proof system. PRF is a pseudorandom functions. G is a generator point on elliptic curve and $xcoord(\cdot)$ returns X coordinate of a point. $PK_V = sk_V G$; $PK_j = \prod_{i=1; i \neq j}^k sk_i G$; $Q = \prod_{i=1}^k sk_i G$.

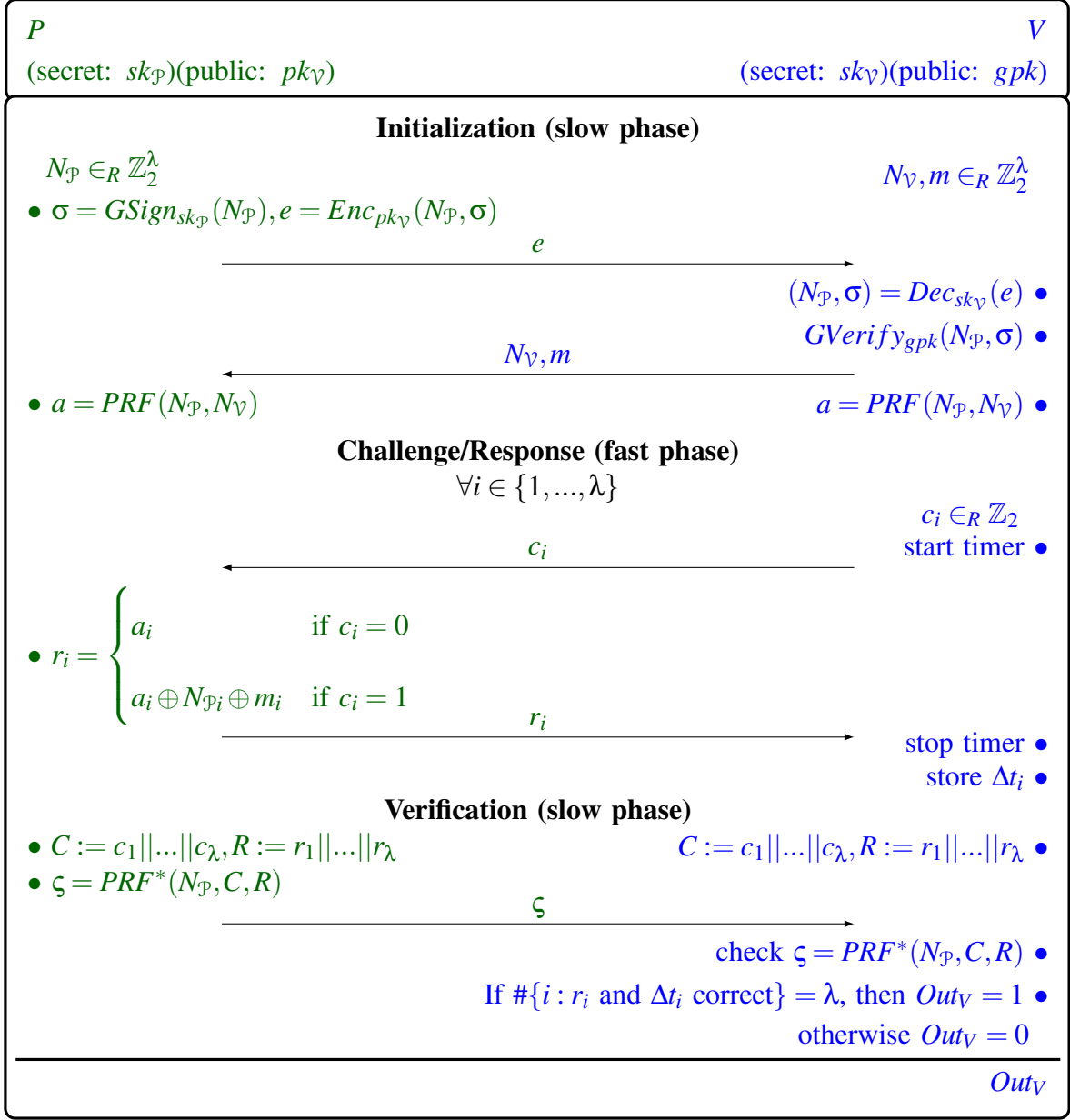


Figure 3.14: SPADE anonymous DB protocol. λ is security parameter. sk_P is prover's private key and gpk is group public key. sk_V is verifier's private and pk_V is verifier's public key. $(GSign_{sk}, GVerify_{gpk})$ is a group signature scheme. (Enc_{pk}, Dec_{sk}) is a secure public-key encryption scheme. $PRF : \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \rightarrow \mathbb{Z}_2^\lambda$ and $PRF^* : \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \times \mathbb{Z}_2^\lambda \rightarrow \mathbb{Z}_2^\lambda$ are pseudo-random functions. λ is the security parameter. N_P and (N_V, m) are nonce values of prover and verifier, and (c_i, r_i) is a challenge and response round.

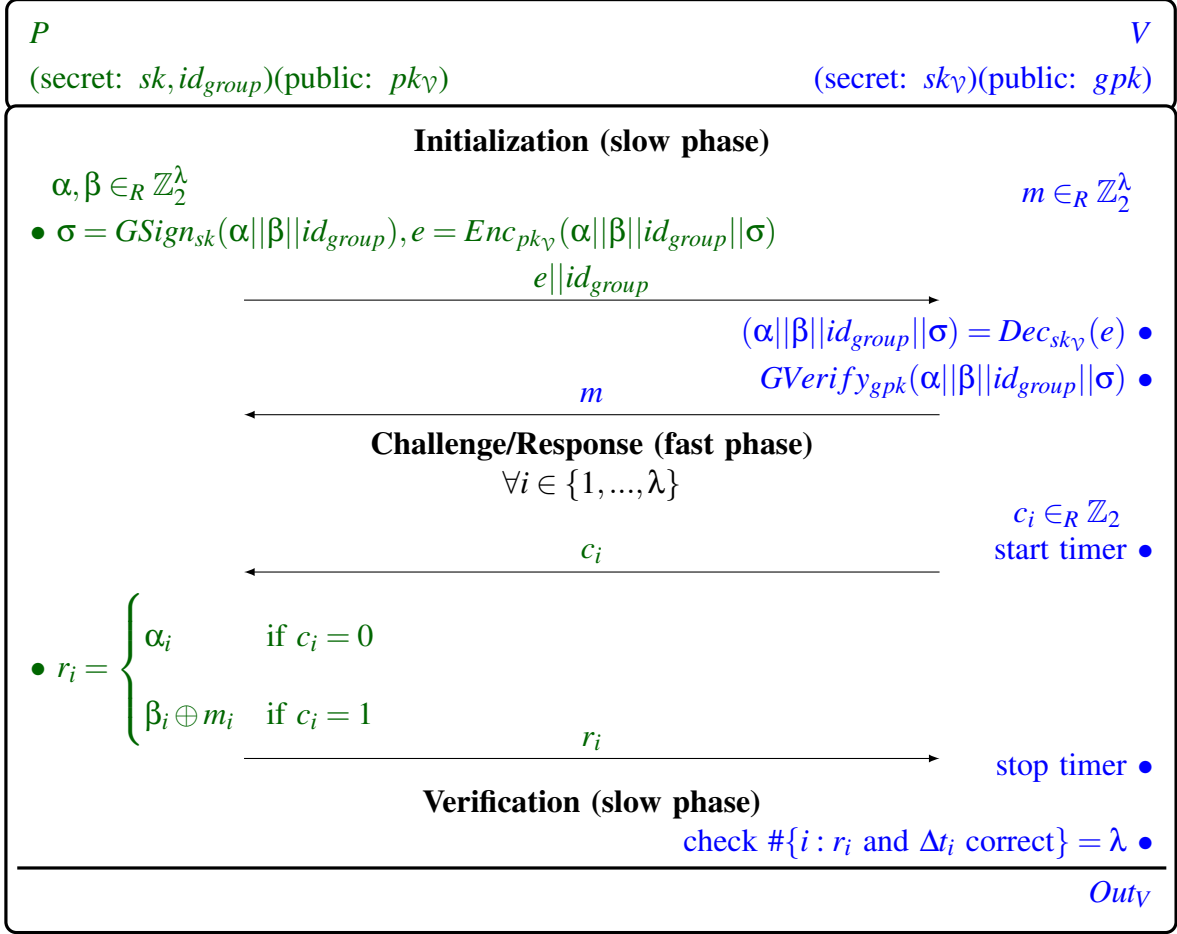


Figure 3.15: TREAD public-key/anonymous DB protocol. λ is security parameter. sk_P is prover's private key and gpk is group public key. sk_V is verifier's private and pk_V is verifier's public key. $(GSign_{sk}, GVerify_{gpk})$ is a group signature scheme. (Enc_{pk_V}, Dec_{sk_V}) is a secure encryption scheme. id_{group} is name of the group that the prover belongs to for anonymous authentication.

3.1.4 Concluding Remarks

In this section we introduced symmetric DB protocols and showed that how the fast phase operations are running on every single bit of the secret. Then we described some important public-key DB protocols and noticed that it is harder to make the fast phase operations depend to the private-key of prover, compared to symmetric key DB protocols. A lot of public-key DB protocols fail to resist against TF attack because of this. There are only two public-key DB protocols [Vau14, ASN14] that

are secure against all DB attacks, which have different public-key settings and have been published at the same time. [ASN14] is a part of this thesis. The model of [Vau14] however, is considered as extension of proof-of-knowledge models, which is not inline with DB attack scenarios, specially MF as a man-in-the-middle attack.

We described all existing anonymous DB protocols. The first anonymous DB protocol that is secure against TF attack is part of this thesis [ASN14]. We later introduce a special TF attack that breaks all of the existing anonymous DB protocols, including [ASN14].

In Table 3.1 we summarize the security properties of the most important DB protocols. Note that the definition of security properties in each paper of this table may be different.

Protocol	Security Status Against Attack			Noise	Anonymous
	<i>DF</i>	<i>MF</i>	<i>TF</i>		
[BC94] P	sec [GAA11]	sec	insec	No	none
[ČBH03] S	sec	sec	insec	No	Eav
[BB04] P	insec [BBM ⁺ 13]	sec	insec	No	Eav
[HK05] S	sec	sec	insec	-	Eav
[RNTS07] S	sec	insec	sec	-	Eav
[TP07] S	sec[MP08]	insec	sec	Yes	Eav
[NV08] S	sec	sec	insec	No	Eav
[KAK ⁺ 08] S	sec	sec	sec	Yes	Eav
[KA09] S	sec	sec	insec	Yes	Eav
[ALM11] S	sec	sec	sec	-	Eav
[FO13b] S	sec	sec	sec	Yes	Eav
[BMV13b] S	sec	sec	sec	Yes	Eav
[HPO13] P	sec	sec	insec	Yes	MiM
[GOR14] A	sec	sec	insec	Yes	none [Vau16]
[Vau14] P	sec	sec	sec	Yes	Eav
[Vau15] P	sec	sec	insec	Yes	Eav
[BGG ⁺ 16] A	sec	sec	insec ↓	Yes	Verifier
[ABG ⁺ 17] A	sec	sec	insec ↓	Yes	Verifier

Table 3.1: Security of different DB protocols. The first column shows the name of protocol and its type **S**ymmetric-key/**P**ublic-key/**A**nonymous. The fifth column shows if the protocol is noise resistant or not. The last column shows who the prover is anonymous to: **E**avesdropper, **MiM**, or **V**erifier. Two colored cells with same color are referring to a single paper. The symbol ↓ denotes this thesis.

3.2 Time-based Entity Authentication

In this section we look at authentication schemes that use time as source of freshness. We can categorize the time-based protocols into two classes by looking at the way they handle the activation of processes and the delivery of messages; In synchronous models, time is captured as a sequence of rounds. In each round all processes are activated simultaneously, and messages are exchanged instantly. In asynchronous models, there is no explicit assumption on the global passing of time, and the focus is on the ordering of events. Process activation is usually message-driven and the adversary controls message delivery and participant activation. The main difference between asynchronous and challenge-response-based models is the level ordering of events. In asynchronous distributed systems there is a form of ordering between different events. On the other hand, in nonce-response-based systems, the ordering is only between challenge and response messages.

Synchronous models are usually adapted when the focus is on a timeliness guarantee, such as termination of a process. However, asynchronous models are taken as better abstraction of real communication systems, as they make no assumptions about network delays and the relative execution speed of the parties, and they nicely capture the view that communications networks are hostile environments controlled by malicious agents.

In this section we review some important models that consider time in entity authentication; Bellare-Rogaway [BR93] proposed the first formal model of entity authentication. Schwenk [Sch14] and Barbosa-Farshim [BF09] extend Bellare-Rogaway to model asynchronous systems using timestamps.

3.2.1 Bellare-Rogaway Model [BR93]

Bellare and Rogaway (BR) [BR93] proposed the first formal model of security for entity authentication, mutual authentication and authenticated key exchange (AKE) protocols. The problems that are considered in this work come in various flavors: there maybe two parties involved or more, the

authentication may be unilateral or mutual, parties may or may not share a secret key.

This model considers two adversarial cases: In the first case, the communication is trusted and it is one of the parties who may be adversarial, and in the second case, the individual parties may be good or bad, but their communication is controlled by the adversary. In the latter case, the adversary can deliver messages out of order and to unintended recipients, and can concoct messages of their own choosing. The adversary can conduct as many sessions as she pleases amongst the parties, and can control, for each, who is attempting to authenticate to whom.

The faithful relaying of messages among the communication partners does not constitute an attack; indeed, the adversary has functioned just like a wire, and may as well not have been there. The idea of mutual authentication is then simple but strong: *a protocol is secure if the only way that an adversary can get a party to accept is by faithfully relaying messages in this manner* (called benign adversary). In order to formalize this simple idea, the main tool in this model is a notion of *matching conversations* (See Definition 2.3.2).

BR provides a general model, in which the communications among interacting parties is under the adversary's control. In particular, the adversary can read the messages produced by the parties, provide messages of her own to them, modify messages before they reach their destination, and delay messages or replay them. Most importantly, the adversary can start up entirely new "instances" of any of the parties, modeling the ability of communicating agents to simultaneously engage in many sessions at once. Each party is modeled by an infinite collection of oracles which the adversary may run.

Note how this differs from the models underlying notions such as interactive proofs [GMR89] or secure function evaluation [Yao82]. In the former case communication is trusted and it is one of the parties who may be adversarial; in the later case, individual parties may be good or bad, but their communication proceeds as the adversary is willing.

The notion of zero-knowledge proof of knowledge [GMR89] has underlined identification pro-

protocols, it does not attempt to model attacks in which responses of entities are played off against one another, as is required for the distributed setting. Furthermore, unilateral authentication is not proving knowledge of a secret insofar as it is fundamentally irrelevant that an agent A knows a in the sense that it can be extracted by a simulator. All the focus in BR is that "*the good party can prove its identity and a bad part can't*".

The formal description of BR model is presented in Section 2.3 at Definition 2.3.1, Definition 2.3.2 and Definition 2.3.3. As defined in Section 2.3, oracle $\pi_{i,j}^s$, $i, j \in I$ denote player i (called initiator) attempting to authenticate to player j (called responder) in session s , where $I \subseteq \{0, 1\}^k$ is the set of identities that defines players. Adversary E communicates with the oracles via queries as defined in Section 2.3. These definitions formalize authentication protocol in a timeless manner. However, BR model is compatible with time.

Modeling Time in BR [BR93]

BR provides an abstract idea of how their model can be compatible with time. In a particular execution of a protocol (Definition 2.3.1), the adversary's i -th query to an oracle is said to occur at time $\tau_i \in \mathbb{R}$. We demand that $\tau_i < \tau_j$ when $i < j$. One notion of time that satisfies this demand includes "*abstract time*" where $\tau_i = i$, or "*Turing machine time*" where $\tau_i =$ the i -th step in E 's computation, when parties are realized by interacting Turing machines. Another notion of time, but a harder to formalize, is "*real time*", where τ_i is the exact time when the i -th query is made, when parties are realized by interacting computers.

3.2.2 Schwenk [Sch14]

The main problem with time is that we cannot model real time (same as so called "physical time" in [Lam78]) in a Turing Machine based model. Instead, [Sch14] considers timestamp to guarantee freshness, and global clocks to provide asynchronous system, that uses interactive Turing Machines model. This model tries to stay as close as possible to the definition for secure authentication

protocols given in BR [BR93].

Schwenk Time-Security

In the seminal paper BR [BR93], the security definition of authentication protocols is motivated by introducing a *benign adversary*, who forwards all messages faithfully. Then they defined an authentication protocol to be secure if *"the winning probability of any adversary is (up to a negligible difference) equal to the winning probability of the benign adversary"*.

They showed that this condition is, for many protocols using random nonces, equivalent to requiring that both parties only accept if they have matching conversations (Definition 2.3.2).

The main goal of Schwenk [Sch14] is to find a replacement for the concept of matching conversations, since in one- and two-message protocols, this concept is not applicable. In one- and two-message protocols, the responder oracle always has a matching conversation to the initiator oracle, but due to replay attacks active adversaries may influence the system significantly: with a benign adversary, there is at most one responder oracle that will accept on a single message; with an active adversary, there may be arbitrary many.

Schwenk considers a protocol to be time-secure, if *"for each initiator oracle that has sent a message there is at most one responder oracle that accepts, and that this responder oracle will accept only if the message was forwarded unmodified by the adversary"*.

This model facilitates an asynchronized mechanism to parties involved in one- or two-message protocols, that are all modeled as Turing machines. Thus there always is one oracle (responder) that has to decide whether to accept or reject after receiving a single message, and before (or without) sending a message.

In a nutshell, in this model time is considered as a global counter that is delivered to a party on request from an special party GB. This is similar to the idea of Network Time Protocol (NTP [BMK10]). If a fresh message has to be sent, the sending TM request a timestamp $ts = (t, aux)$

from GB. Upon reception of such a request, GB first increases its local counter ($t \leftarrow t + 1$). Then the actual value t is returned, optionally with auxiliary data aux appended. This auxiliary data may *e.g.* be a digital signature, to validate the time value.

For two-message protocols, we can additionally base the acceptance condition for initiator oracles (which send and receive one message) on the classical notion of matching conversations, or we can also apply the notion of time-security here.

In the following we formally describe this model;

Schwenk Formal Model

The system consists of multiple parties $\{P_i\}_{i=\{1\dots n\}}$ that may be located in different locations, and process oracles $\{\pi_j^i\}_{j=\{1\dots l_i\}}^{i=\{1\dots n\}}$, where π_j^i is run by party P_i for all $i = \{1\dots n\}$ and $j = \{1\dots l_i\}$. The parties and the process oracles are modelled as Turing Machines. Oracles can be initialized, send and receive messages, and terminate in any of the states $\{\text{finished}, \text{accept}, \text{reject}\}$.

Each party P_i has two local variables: a local timer t_i and a long-term key k_i , that can be either a key pair (*i.e.*, $k_i = (sk_i, pk_i)$), or a list of $n - 1$ symmetric keys (*i.e.*, $k_i = \{k_{i,1}, \dots, k_{i,i-1}, k_{i,i+1}, \dots, k_{i,n}\}$). All the process oracles $\pi_1^i, \dots, \pi_{l_i}^i$ of a process P_i ; have access to the long-term key k_i and can increase the local timer t_i .

The internal states of each process oracle (such as nonces, intermediate states and session keys) are only known to a single oracle, that is described in the following variables:

- the current state of the oracle is stored in the variable Λ ,
- the session key value k is stored in the variable K when the oracle enters to accept or finish states,
- the transcript of all sent and received messages are stored in variable T_j^i (in chronological order).

There is an special party GB with local counter T with value t , that upon receiving a request, increases the value of t by 1 and returns the actual value of t as a message $ts = (t, aux)$ over the network.

The adversary \mathcal{A} is another special party, as a Turing Machine, which implements a strategy to break the cryptographic protocol. The winning conditions of the adversary depends on the security property.

When a process oracle π_j^i receives a time stamp $ts = (t, aux)$ checks the following: it rejects if $t \leq t_i$, otherwise goes through and $t_i \leftarrow t$. When π_j^i wants to send a message to P_k , it request a time stamp $ts = (t, aux)$ from GB, and if $t > t_i$ then $t_i \leftarrow t$ gets used in the message to P_k . The time stamp doesn't need to be authentic if the adversary has complete control on the network, however if the adversary only has partial control on the network, the authentic time stamp is useful.

This time model is used in the first message of a protocol. The ordering of the following messages of the protocols is checked by nonces.

This model provides an ordering system that updates the local clock of parties with the help of a third party GB. The only event that causes the parties (through their process oracle) to update their local clock, is receiving a message with higher clock value.

The disadvantage of this model is that it adds a new global party that needs to generate and transmit a new timestamp for every single message that is sent by the communicating parties.

3.2.3 Barbosa-Farshim [BF09]

Since it's easier to formalize asynchronous models, such as the ones described earlier in this section, they are much more used. This trend however, comes at the cost of abstracting away many of the practical uses of time-variant parameters in cryptographic protocols, which rely on explicit representations of time.

The security of timestamp based mechanism relies on the use of a common time reference. This means that each party must have a local clock that must be synchronized to an extent with others in order to accommodate the acceptance window. The local clocks must also be secure to prevent adversarial modification: if an adversary is able to reset a clock backwards, then it might be able to restore the validity of old messages; conversely, by setting a clock forward, the adversary might have advantage in preparing a message for some future point in time. These assumptions on the security and synchronization of local clocks may be seen as disadvantages of using timestamps, since in many environments they may not be realistic.

In Barbosa-Farshim [BF09], a general approach is presented in modeling such system to permit analyzing protocols relying on timestamps. They introduce two different models (one based on Canetti-Krawczyk [BCK98] and one based on Bellare-Rogaway [BR93] entity authentication models) to model AKE with timestamps.

Barbosa-Farshim models time as a local clock, which is incremented by sending TICK requests. To preserve the common asynchronous trait in these models, where the adversary controls the entire sequence of events occurring during an execution, Barbosa-Farshim does not allow the clocks to progress independently. Instead, they leave it to the adversary to control the individual clocks of parties: the adversary can query Tick (or activation) through which it can increment the internal clock of an honest party (of course it has complete control of the clocks of corrupted parties). The adversary is not allowed to reset or cause the internal clocks to regress in any way. This restriction captures the real-world assumption, in which the internal clocks of honest parties must be, to some extent, secure.

The addition of these elements to the BR and CK models allows to capture the notion of time and internal clock drifts, while preserving the asynchronous nature of those models by allowing the adversary to freely control the perception of time passing at the different parties.

Barbosa-Farshim Replay Resistance

In protocols that use timestamp to prevent replay attacks, the receiver defines an acceptance window and temporarily stores received messages until their timestamps expire. The width of the acceptance window must be defined as a trade-off between the required amount of storage space, the expected message transmission frequency, speed and processing time; and the required synchronization between the clocks of sender and receiver. Received messages are discarded if they have invalid timestamps, or if they are repeats within the acceptance window.

In this setting, the local clocks are not explicitly used to keep track of elapsed time, but simply to ensure that the receiver does not have to store all previously received messages to prevent accepting duplicates. In fact, for this purpose, timestamps are essentially equivalent to sequence numbers. Furthermore, synchronization of clocks between sender and receiver is less of a timeliness issue, and more of an interoperability problem.

For example, two honest parties using this mechanism might not be able to communicate at all, even without the active intervention of any adversary, should their clocks values be sufficiently apart. This is reminiscent of a Denial-of-Service attack, which is usually out of the scope of cryptographic security analyses. Although the adversary may be able to prevent successful completions of protocols (*e.g.* by driving internal clocks significantly out of synchronization, or simply by not delivering messages), but in consistence with the original models, the security definitions for cryptographic protocols using timestamps in this context remain unchanged.

Barbosa-Farshim Time-Security

For protocols that use timestamps to obtain timeliness guarantees on messages, the local clock values are taken for what they really mean: time measurements. The receiver of a message may require assurance that an accepted message was generated recently with respect to its own local clock, where recently is quantifiable as a time interval.

In order to provide these guarantees, accuracy of internal clocks of the honest parties in the system must be considered. In this work, a limit on the maximum pair-wise drift that the adversary can induce between the internal clocks of different parties is imposed (see Definition 3.2.1). In this modeling approach, a protocol with timeliness-security must guarantee that any adversary breaking this requirement must be overstepping its maximum drift allowance with overwhelming probability.

Definition 3.2.1 (δ -synchronization) *An adversary satisfies δ -synchronization if it never causes the clock variables of any two (honest) parties to differ by more than δ .*

This definition captures the notion that clocks must be synchronized in order to achieve any sort of timeliness guarantee. We are now in a position to state Barbosa-Farshim definition of entity authentication that captures timeliness too.

Let π_A^i and π_B^j be two partner oracles where the latter has terminated. Also, let $t_B(E)$ be the function returning the value of the local clock at B when event E occurred. Finally, let $acc(A, i)$ denote the event that π_A^i accepted, and let $term(B, j)$ denote the event that π_B^j terminated.

Definition 3.2.2 (β -Barbosa-Farshim Authentication (β -BFA)) *A key exchange protocol provides β -BFA if it provides initiator-to-responder authentication, and furthermore for any honest responder oracle π_B^j which has terminated with partner π_A^i , for honest A , we have:*

$$|t_B(term(B, j)) - t_B(acc(A, i))| \leq \beta$$

3.2.4 Concluding Remarks

Formalizing time in distributed systems is a difficult task and it is not possible to model real time with Turing machine based models [Sch14]. This made most of the researchers to formalize the concept of timestamp, as a limited notion of time, that provides ordering to the events of a distributed system and allows to guarantee freshness of messages without using challenge-response. In this research we are interested in *synchronization* of participants as an important property of

time. This property is defined in different models and some implementations have been provided to facilitate it.

In this thesis, we extend these models in order to add location synchronization (proximity) to propose a new approach in distance-bounding.

3.3 Other Literature Reviews

In Chapter A we review some other literatures that are related to our research. First we describe some the important symmetric key distance-bounding schemes in Section A.1 and then describe difficulties and limitations of implementing accurate distance-bounding systems in Section A.2. Then in Section A.3 we describe the notions of privacy that are considered in distance-bounding.

Chapter 4

Public-Key Distance-Bounding

In this chapter we first show that if protocol participants have access to a directional antenna, many existing protocols that have been proven secure, will become insecure, and then propose a model to include this new capability of the users (called DBID). This approach provides a natural way of modeling man-in-the-middle attack in line with identification protocols, as well as other attacks that are commonly considered in distance-bounding protocols. We propose a new DBID scheme, called Poxy, with security proof. We compare the existing public key DB models, and prove the security of the scheme known as ProProx, in our model.

Recently public key DB protocols have been proposed where the prover is only known through their public keys, while their secret key remains private to them [ASN14, Vau14, ASN17a]. In these models, the verifier only has access to system public parameters as well as public keys of the participants.

In a DB setting there are three types of *participants*: *provers* who are registered in the system and have secret keys, a *verifier* who is honest and has access to correct public keys of provers, and *actors* who are not registered in the system, but want to be accepted and may collude with a dishonest prover. The distance between the prover and the verifier is measured by using a “fast challenge-response phase” during which a sequence of one bit *challenges* are sent by the verifier to the prover, and the corresponding responses by the prover is recorded and used for distance estimation. A *challenge-response table* includes responses that are required for all possible challenges and is

calculated by the prover before the fast challenge-response rounds start. The challenge-response table is constructed using the provers' secret key, and some nonces that are communicated during the slow phase of the protocol.

In symmetric key setting, the challenge-response table can also be constructed by the verifier and used for the verification of responses. In public key setting however, the verifier only knows the prover's public key, and cannot calculate the challenge-response table. In this case, the verifier verifies the correctness of the prover's responses using their relation with the provers' public key.

For a DB protocol with distance bound \mathcal{B} , we refer to participants whose distance to the verifier are less than \mathcal{B} as *close-by* participants (set \mathcal{S}) and those who are farther away than \mathcal{B} , as *far-away* participants (set \mathcal{F}).

To prove security of the existing public key DB protocols, such as [ASN14, BC94, BB04, Vau14, GOR14, ASN17a, GKL⁺14], PoPoK [Vau14] proposed a formal security model that uses a cryptographic proof-of-knowledge system (Definition 2.5.3) and considers distance bound as an additional property of the system. In DBID [ASN17a] an alternative approach was proposed that follows the security formalization of identification schemes (Definition 2.4.1) with the framework of Σ -protocols (Definition 2.4.3), and includes distance-bound as an extra property. The ProProx scheme [Vau14] was first proven secure in the former model [Vau14], and later in the latter model [ASN17a].

ProProx uses polynomial times more fast phase operations, compared to normal DB protocols. The communications of the fast phase of DB protocols are generally more expensive, less reliable and more noise sensitive compared to the slow phase, as the data is sent in plain form. This fact makes the ProProx protocol to be an inefficient scheme. DBPoK- \log^+ [ASN14] is another public key DB protocol that uses a different cryptosystem and uses the normal amount of fast phase communications, which makes it more efficient compared to ProProx. However, the security proof of this protocol has not been yet provided and it is not reliable in presence of noisy channel.

Our work: This chapter describes the last state of our research in public-key distance-bounding, which includes three publications [ASN14, ASN17a, ASN17b]. We consider provers that have access to directional antennas. Such antennas allow point to point communication with minimum interception by eavesdroppers who are outside the main transmission direction [ARS16]. Advances in beamforming techniques and smart antennas in recent years [ARS16] have made these antennas readily accessible to users. Distance bounding protocols, during the fast challenge-response phase, rely on physical layer communication and so it is important to consider this extra attacking capability for protocol participants.

We will show that indeed directional antenna affects the security evaluation of DB protocols, and in particular effectively allows a malicious prover to launch a successful TF attack against protocols that had provable security against this attack. In Section 4.1 we show how this extra capability can be used by a malicious prover who is aided by a helper to break security of VSSDB [GKL⁺14] and DBPoK- \log^+ [ASN14] schemes. Directional antennas had been previously considered for actors during MF attack. In this paper we consider a dishonest prover with access to this type of antenna. For distance fraud, a directional antenna does not appear to affect security. In TF however, the dishonest prover is aided by a helper and directional antenna and this affects the security definition.

We extend the DBID formal security model [ASN17a] to include this new attacker’s capability. The directional TF attack is captured in the revised TF-resistance (Property 4.2.4). We propose a new DBID scheme, called Poxy, and provide the security proof. We also prove that the existing ProProx scheme is indeed secure in this new model.

Organization. Section 4.1 shows directional TF attack on a public key DB protocol. Section 4.2 presents our model, Section 4.3 and Section 4.4 describe the construction of Poxy and ProProx, respectively, and give security theorems and proofs. Section 4.5 gives a summary of related works, and Section 4.6 concludes the chapter.

4.1 Directional Attacks on Public-Key DB Protocols

Directional attacks assume that participants have access to directional antennas that allow them to direct messages to specific participants, and prevent other participants from receiving them. Figure 4.1 shows how such an antenna can be exploited by a malicious prover in a TF attack. The helper does not receive slow phase messages that are sent by the prover, as prover uses a directional antenna (orange ribbon in Figure 4.1) for communication in this phase. Before the start of the fast-phase, the prover sends all fast-phase responses (*e.g.* , the fast challenge-response table) to the helper, making the helper in-charge of responding to the fast-phase challenges.

This means that the adversary is able to separate the slow phase messages of the protocol from the fast-phase messages. In a vulnerable protocol, the prover may succeed in TF attack without leaking their long term key to the helper, using this separation technique. Therefore, the attacker's success in TF will not imply success in future impersonation.

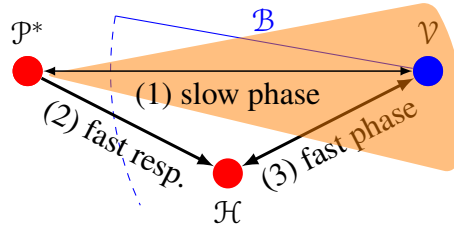


Figure 4.1: Directional TF

In the following we describe how this setting helps a malicious prover to succeed in terrorist-fraud against VSSDB [GKL⁺14] and DBPoK-log⁺ [ASN14].

Attack against VSSDB [GKL⁺14]

Using Definition 4.2.1 for a DB scheme, Figure 3.11 presents the Π protocol (Definition 4.2.1) of VSSDB scheme. This is a protocol between the prover and the verifier where the prover has access to the public key of the verifier and their own secret key, and the verifier has access to their private key and the public key of the prover.

Lemma 1 *In the Π protocol of VSSDB scheme (Figure 3.11), the fast challenge-response table does not leak information about the secret value $sk_{\mathcal{P}}$ of prover, assuming that $sk_{\mathcal{P}}$ and x are independently chosen.*

Proof 1 *The elements of the fast challenge-response table are calculated as $r_j = f_j(c'_j) \in \{0,1\}$ for $j = \{1, \dots, \lambda\}$. Therefore, by knowing the table, one can, at the most, extract the values of e_j, k_j, l_j for $j = \{1, \dots, \lambda\}$. By finding these values, one can extract the value of x using the equation $x_j = e_j \oplus k_j \oplus l_j$ for $j = \{1, \dots, \lambda\}$. Since k_j and l_j are chosen randomly, therefore, this table only contains information about randomly chosen values k and l , and the value of x , which are independent of the secret value $sk_{\mathcal{P}}$. ■*

Attack 4.1.1 *In this attack, the prover sends the messages of the slow phase (i.e., $N_{\mathcal{V}}, M, c_{\mathcal{P}}, \pi, \zeta, \phi$) to the verifier using directional antenna. The prover then sends the fast challenge-response table (i.e., $\forall j \in \{1, \dots, \lambda\} : \text{either } (e_j, k_j) \text{ or } (k_j \oplus l_j, e_j \oplus l_j)$) to the helper before running the fast phase. Note that the fast challenge-response table does not leak the prover long-term secret $sk_{\mathcal{P}}$ according to Lemma 1.*

This allows the helper to respond to the verifier's challenges during the fast phase. The collusion of the prover and the helper will make the verifier to accept (i.e., $Out_{\mathcal{V}} = 1$) and this is without the prover sending to the helper any information that is dependent on the secret key $sk_{\mathcal{P}}$. The secret $sk_{\mathcal{P}}$ is required to generate a valid signature σ in the message π . This means that the helper's success chance in a future impersonation attack will not improve. This completes a successful TF.

Attack against DBPoK- \log^+ [ASN14]

The presented model of [ASN14] follows the original definition of TF (Attack 3.1.3), and so our attack can be seen as outside their model. In this section we present a TF attack against DBPoK- \log^+ , using the more recent definition of TF (Attack 3.1.3).

The Π protocol in DBPoK- \log^+ scheme consists of the following four sub-protocols between the

verifier (V) and the prover (P). The prover takes secret-key (sk_i, r) as input, and the verifier takes prover's public-key $pk_i = g_1^{sk_i} \cdot g_2^r$ as input. The following is the scheme presented in [ASN14], slightly modified to become noise resistant.

Step (i) **Bit Commitment** is a commitment protocol, in which the prover uses the secret key sk_i as input. In this protocol, the prover decides on the "fast challenge-response table" and commits to each bit in the table. The verifier learns the committed values of every single bit of the fast challenge-response table. For security parameter λ , this table consists of two rows: $\{r_b[l]\}_{l=\{1,\dots,\lambda\}, b \in \{0,1\}}$, where $r_b[l]$ is the response in the i^{th} fast challenge-response round. The corresponding committed values are two vectors C_0 and C_1 where $C_b = (C_b[1] \dots C_b[\lambda])$ for $b = \{0, 1\}$, and the corresponding randomness of commitments are indicated by $\{v_b[l]\}_{l=\{1,\dots,\lambda\}, b \in \{0,1\}}$, where $v_b[l] \in \mathbb{Z}_p^*$. The commitment values are calculated as follows: $C_b[l] = g_1^{r_b[l]} \cdot h^{v_b[l]}$ for $b \in \{0, 1\}$, $l = \{1 \dots \lambda\}$, and $g_1, h \in \mathbb{Z}_p$. The committed table and the randomness is kept secret at the prover, while the commitments are sent to the verifier. Figure 4.2 shows the details of this step. The parts that are shown in a box, are sub-protocols whose details are omitted.

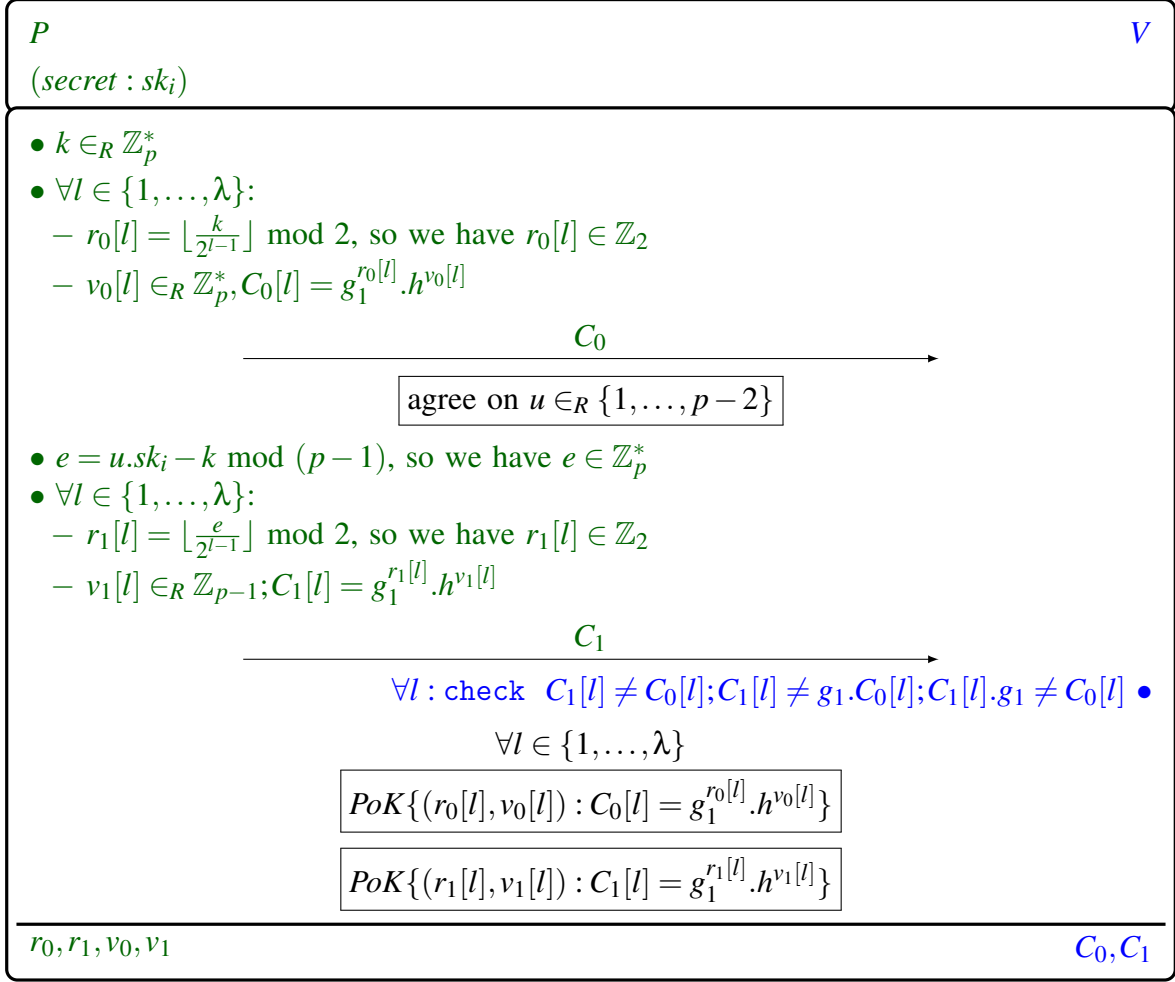


Figure 4.2: Step (i). DBPoK- \log^+ Bit Commitment. r_0 and r_1 form the response table.

Step (ii) Fast Challenge/Response is the protocol in which the prover uses the calculated "fast challenge-response table" $\{r_b[l] : l = \{1 \dots \lambda\}, b = \{0, 1\}\}$, generated in Bit Commitment step, as input. They run the protocol in Figure 4.3.

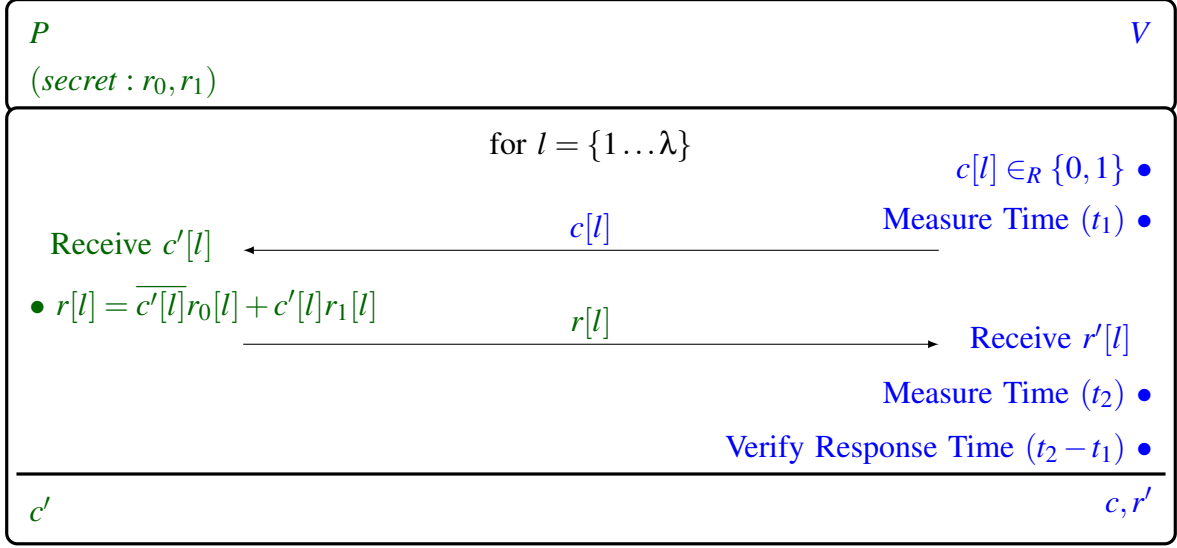


Figure 4.3: Step (ii). DBPoK- \log^+ Fast Challenge/Response

Step (iii) Commitment Opening is used to open half of the commitments, that correspond to the challenge bits sent by the verifier in Fast Challenge/Response step. In this step, the prover uses the secret commitment randomness (*i.e.*, $\{v_b[l] : l = \{1 \dots \lambda\}, b = \{0, 1\}\}$) and the challenge values of Fast Challenge/Response step (*i.e.*, c'). The verifier uses the committed values (*i.e.*, $\{C_b[l] : l = \{1 \dots \lambda\}, b = \{0, 1\}\}$) and the challenge and response values of step (iii) (*i.e.*, c and r') as input. This protocol is shown in Figure 4.4, which improves the original PDB protocol [ASN14] by adding noise resistance to the protocol. This step succeeds, if the noise counter is less than the threshold (*i.e.*, $count_{noise} < \tau$).

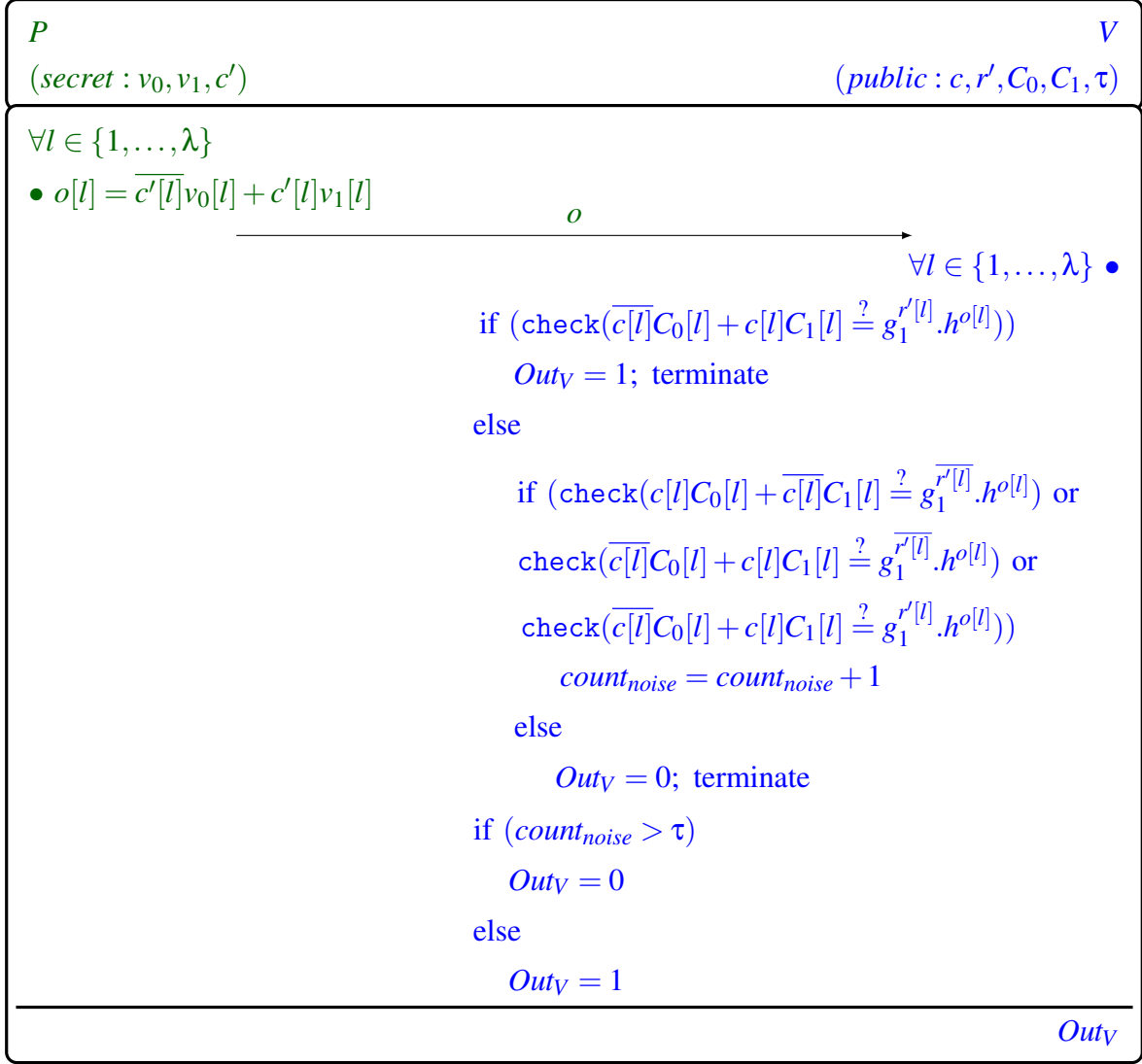


Figure 4.4: Step (iii). DBPoK-log⁺ Commitment Opening

Step (iv) Proof-of-Knowledge is a protocol for zero-knowledge proof of equality that shows the secret key and the bitwise committed secret key of Bit Commitment step are the same. In this protocol, the prover uses the secret key (sk_i, r) and the commitment randomness of Bit Commitment step, and the verifier uses the public-key and bit committed values of Bit Commitment step, as input. z is the accumulation of the committed values of Bit Commitment step as $z = \prod_{l=1}^{\lambda} (C_0[l]C_1[l])^{2^{l-1}} \bmod p$, and v is the accumulation of the commitment randomness of step (ii) as $v = \sum_{l=1}^{\lambda} 2^l \cdot (v_0[l] + v_1[l]) \bmod (p-1)$.

For security parameter t , this protocol runs t iterations of zero-knowledge proof-of-knowledge (Definition 2.5.4 and Definition 2.5.3) where z and pk_i satisfy the following relation: $PoK[(sk_i, v, r) : z = g^{u \cdot sk_i} \cdot h^v \wedge pk_i = g^{sk_i} \cdot g_2^r]$.

If all steps terminate successfully, then the verifier outputs $Out_V = 1$.

Lemma 2 *In DBPoK- \log^+ scheme, the fast challenge-response table (i.e., r_0 and r_1) does not leak any information about the randomness (r) of the secret-key of the prover, except with negligible probability.*

Proof 2 *We know that by having the fast challenge-response table, we can calculate part of the secret-key of the prover, as $sk_i = \frac{k+e}{u} \bmod (p-1)$ where k is fresh randomness. Note that the fast challenge-response table is the output of the random function that takes sk_i as input. So it cannot leak any information about other independent secrets of the prover, including r .*

If there is an adversary \mathcal{A} that can calculate the randomness r from the secret-key sk_i and $pk_i = g^{sk_i} \cdot g_2^r$, then \mathcal{A} can solve the discrete log problem for g_2^r . Therefore, since we assume discrete log is a hard problem, then the success chance of \mathcal{A} is negligible. ■

Attack 4.1.2 *In directional TF attack (Figure 4.1), a malicious far-away prover will use a directional antenna for the slow phase of DBPoK- \log^+ protocol (all steps except step (ii)) to communicate directly with the verifier, without the helper being able to intercept the messages. The prover sends the fast challenge-response table to the helper before running step (ii). Note that the fast challenge-response table does not leak any information about r , according to Lemma 2. In this way, the helper can respond in time and correctly to the challenges of the verifier during the fast challenge-response rounds. This attack makes the verifier to accept the protocol.*

Since the fast challenge-response table does not leak any information about the randomness r , the helper will not be able to pass step (iv) in future and so it cannot impersonate the prover. This completes a successful terrorist fraud on DBPoK- \log^+ .

4.2 Model

First we define the settings of our system. This includes entities, their communication, their views, and the adversarial capabilities. Then we define distance-bounding identification scheme (DBID) and describe DBID experiment, which simulates an instance of DBID scheme. Finally we formalize four properties: (*Completeness*, *Soundness*, *DF-resistance*, and *TF-resistance* of distance-bounding identification schemes, using a game-based approach and described as a DBID experiment where adversary is active, and the game is between a challenger that sets up the system, taking into account the adversary's input. Finally in fundamental lemma, we discuss the relation between the location of participants, the timing of messages and their content.

Entities. We consider a set \mathcal{U} of users. The user $u \in \mathcal{U}$ can have multiple provers that are denoted by the set \mathcal{P} . This captures the scenario that a single user has multiple devices.

A trusted group manager generates the public parameters of the system, and registers users and issues a key pair to each user. The user u is identifiable by its' private key. The private key, that must be kept secret, forms the secret input of the user in providing authentication proof. The private key of a user u is shared by all their provers \mathcal{P} . The corresponding public key of the user is published by the group manager.

There is a single verifier in the system, that for uniformity of notations, we refer to it as a set \mathcal{V} that has a single member. The verifier only access to the public parameters of the system.

There is a set of *actors* (\mathcal{T}) that only have access to the public parameters of the system. In this paper we refer to the members of the sets \mathcal{P} , \mathcal{V} and \mathcal{T} as *participants*.

Each participant has a location $loc = (x, y) \in \mathbb{R} \times \mathbb{R}$, that is an element of a metric space equipped with Euclidean distance, and is fixed during the protocol. The distance function $d(loc_1, loc_2)$ returns the distance between two locations. Message travel time between locations loc_1 and loc_2 is $\frac{d(loc_1, loc_2)}{c}$, where c is the speed of light. A bit sent over the channel may flip with probability

p_{noise} ($0 \leq p_{\text{noise}} \leq 1$).

Participants that are located within a predefined distance bound \mathcal{B} from the verifier, excluding the verifier, are called *close-by* participants (set \mathcal{S}), and those who are outside the distance bound from the verifier are called *far-away* participants (set \mathcal{F}).

Communication Structure. All participants have access to directional antennas: a participant A in loc_A can send a message to participant B at loc_B , such that others who are not in the conic space centered by the straight line connecting loc_A to loc_B , cannot intercept it. Using omni-directional antenna however allows a message to be seen and modified by other participants. A participant may have multiple antennas that can be either directional or omni-directional. We allow a participant to send multiple messages to multiple parties at the same time, each from a separate antenna. Multiple messages that are received at the same time on the same antenna are combined and received as a single message.

View. The view of an entity at a point of a protocol consists of: all the inputs of the entity (including random coin tosses) and the set of messages that they have received up to that point in the protocol. Receiving a message is called an *event*. $View_x^\Gamma(e)$ is a random variable that denotes the view of an entity (or a set of entities) x right after the event e in protocol Γ . The short notation $View_x^\Gamma$ is used to indicate the view of x at the end of the protocol Γ , i.e., $View_x^\Gamma = View_x^\Gamma(e_{\text{last}})$ where e_{last} is the last event in the protocol Γ .

Adversary. An adversary can corrupt a subset of participants $\mathcal{X}^* \subset \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$. As we will see later in this section, for each security property, \mathcal{X}^* will have certain restrictions;

in *Completeness* $\mathcal{X}^* = \emptyset$, in *Soundness* $\mathcal{X}^* \subseteq \mathcal{T}$, in *DF-resistance* $\mathcal{X}^* \subseteq \mathcal{P}$, and in *TF-resistance* $\mathcal{X}^* \subseteq \mathcal{P} \cup \mathcal{T}$.

When a prover of a user u is compromised, the user u 's secret private key is compromised and the adversary can choose devices with that key at locations of their choice. In other words, all the provers in \mathcal{P} become compromised. This is because all the provers of a user share the same

private key. We refer to them as *corrupted provers*, who are controlled by the adversary and may be activated simultaneously. However, we assume the non-corrupted provers follow the protocol, and a user only uses one of its devices at a time (*i.e.*, the execution time of the provers \mathcal{P} do not overlap). This is because an honest user does not use multiple devices simultaneously.

Definition 4.2.1 (Distance-Bounding Identification Scheme) *A distance-bounding identification scheme (DBID) for security parameter λ , is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathbb{P}, \mathcal{B}, p_{\text{noise}}, \text{Init}, \text{KeyGen}, \Pi, \text{Revoke})$, where*

(I) \mathbb{X} and \mathbb{Y} are the sets of possible master keys and public keys of the system, respectively, chosen based on the security parameter λ . The system master key $msk \in \mathbb{X}$, and group public key $gpk \in \mathbb{Y}$ are generated using

$$(msk, gpk) \leftarrow \text{Init}(1^\lambda) \text{ algorithm};$$

(II) \mathbb{S} and \mathbb{P} are sets of possible private keys and public keys of the users respectively, chosen according to the security parameter λ . The user private key $sk \in \mathbb{S}$, and public key $pk \in \mathbb{Y}$ are generated using either $(sk, pk) \leftarrow \text{KeyGen}(1^\lambda, msk, gpk)$ algorithm or $\text{KeyGen}\{U(1^\lambda, gpk) \leftrightarrow \text{GM}(1^\lambda, msk)\}$ protocol;

The KeyGen algorithm is run by the group manager and the output is a user key pair and updated group public key. The user key pair is securely sent to the user, and the public key is published by the group manager, *i.e.*, $gpk' := gpk \cup \{pk\}$. However, the KeyGen protocol is run between the group manager $\text{GM}(1^\lambda, msk)$ and a user $U(1^\lambda, gpk)$. The user outputs a key pair (sk, pk) , and the group manager outputs the updates group public key $gpk' := gpk \cup \{pk\}$.

(III) Π is a Σ^* -protocol between a prover $P(sk, pk, gpk)$ and the verifier $V(pk, gpk)$, in which V verifies if the prover is authentic and is located within the distance bound $\mathcal{B} \in \mathbb{R}$ to the verifier.

(IV) The transmitted bits of a fast challenge-response round in Π protocol are affected by noise

where $p_{\text{noise}} \in [0, 1]$ is the probability of a bit flip on each fast challenge-response message.

(V) $(gpk') \leftarrow \text{Revoke}(msk, gpk, i)$ is an algorithm that takes the master secret key, the group public key and the index of a user. The algorithm removes the corresponding user u_i from the system and updates the group public key accordingly, i.e., $gpk \rightarrow gpk'$. The Revoke operation is optional in DBID scheme.

Below we describe execution of an instance of the DBID scheme, which we call DBID experiment.

Definition 4.2.2 (DBID Experiment) A DBID experiment is defined by a tuple $(\text{DBID}; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T})$, where

- (i) DBID is a distance-bounding identification scheme as defined in Definition 4.2.1.
- (ii) \mathcal{U} is the set of users that are members of the group; each user $u_j \in \mathcal{U}$ has three attributes:
 - $u_j.\text{Key}$ that is a secret key generated by the group manager,
 - $u_j.\text{RT}$ that is the registration time of the user that can be any time, and
 - $u_j.\text{Rev}$ that is a flag that shows if the user is revoked.
- (iii) \mathcal{P} is the set of provers; each prover has access to the secret key of a single user.
- (iv) \mathcal{V} is the set of verifiers; that have access to the public parameters of the DBID system. We consider the case where \mathcal{V} has a single member.
- (v) \mathcal{T} is the set of actors; each actor has access to the public parameters of the DBID system.

Members of the set $\mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ are called participants of the system. Each of the participants $x \in \mathcal{X}$, has the following attributes:

- a1. $x.\text{Loc}$ is the location of the participant,
- a2. $x.\text{Code}$ is the code to run by the participant,
- a3. $x.\text{St}$ that is the start time of the $x.\text{Code}$ execution, and

a4. $x.Corr$ is a flag indicating if the participant is corrupted or not.

In addition to these attributes, each prover $p \in \mathcal{P}$ has one extra attribute:

a5. $p.Key$ that is the secret key of the corresponding user, i.e., $p.Key = u_j.Key$ for user $u_j \in \mathcal{U}$.

The start time of all provers is after registration time of the user, i.e., $\forall u \in \mathcal{U}, \forall p \in \mathcal{P} : p.St > u.RT$.

The provers of a user are either all honest or all dishonest. Because of users' keys are independently chosen, we can only consider a single user and so for simplicity we omit other users. i.e., $\forall p \in \mathcal{P} : p.Corr = flag$, where $flag \in \{true, false\}$. Honest provers $p \in \mathcal{P}$ follow the Π protocol (i.e., $p.Code = DBID.\Pi.P(.)$) and there is no overlap in the execution time of the honest provers. If the verifier is honest, then it follows the Π protocol (i.e., $v.Code = DBID.\Pi.V(.)$ for $v \in \mathcal{V}$).

The experiment is run by a simulator that sets the attributes of the participants, and interacts with the group manager to assign keys to the provers of a user. If there is an adversary in the system, the simulator interacts with the adversary and follow their requested operations, that will influence the experiment.

The experiment, without an adversary, proceeds as follows:

1. Setup.

(a) **Initialize:** The group manager runs $(msk/gpk) \leftarrow DBID.Init(1^\lambda)$ algorithm to generate the master secret key and group public key.

(b) **Generate Players:** The simulator forms the sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ and sets their attributes. The simulator interacts with the group manager obtain and assign keys of the provers.

2. **Run:** The simulator starts the execution of $x.Code$ for all participants $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ at time $x.St$.

The simulation uses a clock. $Time(e)$ indicates the time of event e . The start and finish time of a protocol Γ is indicated as $stTime(\Gamma)$ and $fshTime(\Gamma)$ respectively, which form the execution

time $exTime(\Gamma) = (stTime(\Gamma), fshTime(\Gamma))$ as the range of time and the execution time period $exLen(\Gamma) = fshTime(\Gamma) - stTime(\Gamma)$. Different provers have different execution time period (*i.e.*, they participate in a protocol from time t_1 to t_2), and possibly different locations.

In the following, we define security properties of DBID scheme, using a game between a challenger and an adversary. This game is a DBID experiment that is run by the challenger who interacts with an adversary. In this game we only consider one user, *i.e.*, $|\mathcal{U}| = 1$. The challenger plays both roles of the simulator and the group manager in the DBID experiment (Definition 4.2.2). The adversary's capabilities is modelled as access to a query that it presents to the challenger.

Definition 4.2.3 (DBID Game) *A DBID game between a challenger and adversary is a DBID experiment that is defined by a tuple $(DBID; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T}; \text{CorruptParties})$ where*

- *DBID is a distance-bounding identification scheme as defined in Definition 4.2.1.*
- *$\mathcal{U}, \mathcal{P}, \mathcal{V}, \mathcal{T}$ are the sets of users, provers, verifiers and actors as defined in Definition 4.2.2, that are determined through interaction of the challenger and the adversary.*
- *$\text{CorruptParties}(Q)$ is a query that allows the adversary to plan (program) their attack. Q is a set of participants, that may exist in the system or be introduced by the adversary.*

The game setup phase is by the challenger while playing the roles of the simulator and the group manager, and interacting with the adversary. In more details:

1. Setup:

(a) **Initialize:** *Challenger runs $(\text{msk}/\text{gpk}) \leftarrow \text{DBID.Init}(1^\lambda)$ and publishes gpk . Note that the execution codes of an honest prover and verifier are known by the challenger and the adversary at this point, and are referred to as $\text{DBID}.\Pi.P$ and $\text{DBID}.\Pi.V$, respectively.*

(b) **Generate Players:** *The sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ are formed through the interaction of the challenger and the adversary as follows:*

i. The challenger creates the sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ as follows:

- Chooses a verifier $\mathcal{V} = \{v\}$, with the following attributes:

a1. $v.Loc = loc_0$,

a2. $v.Code = \text{DBID}.\Pi.V$,

a3. $v.St = 0$, and

a4. $v.Corr = \text{false}$.

- Runs $(sk, pk) \leftarrow \text{DBID.KeyGen}(1^\lambda, msk, gpk)$ once and forms the set $\mathcal{U} = \{u\}$. The user key is set as $u.Key = sk$, the registration time of the user is set as $u.RT = 0$ and the revocation flag is set as $u.Rev = \text{false}$. The group public key is updated as $gpk' := gpk \cup \{pk\}$.

- Creates a prover set \mathcal{P} and for each member p of \mathcal{P} , assigns their attributes as:

a1. $p.Loc$ is set arbitrarily,

a2. $p.Code = \text{DBID}.\Pi.P$,

a3. $p.St$ is set arbitrarily such that there is no overlap in the execution time of the provers (i.e., $\nexists p_1, p_2 \in \mathcal{P} : p_1.St < p_2.St \wedge p_1.St + \text{exLen}(\text{DBID}.\Pi) > p_2.St$),

a4. $p.Corr = \text{false}$, and

a5. secret key $p.Key = u.Key$.

- $\mathcal{T} = \emptyset$

ii. The challenger sends the attributes $(x.Loc, x.Code, x.St)$ for all $x \in \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ to the adversary. The size of the set \mathcal{X} is n .

iii. The adversary forms the corruption query $\text{CorruptParties}(Q)$ using the published values, and sends it to the challenger. The secret information of the corrupted partic-

ipants in Q is given to the adversary and the behaviour (Code) of the corresponding participants, is assigned according to the adversary instruction.

More specifically, the parameter of this query is $Q = \{q_1, \dots, q_{n'}\}$. Each q_i consists of the location, the execution start time and the execution code of a participant. i.e., $q_i = (\text{type}, \text{location}, \text{code}, \text{time})$, where $\text{type} \in \{\text{verifier}, \text{prover}, \text{actor}, \text{user}\}$ indicates the type of the participant, $\text{location} \in \mathbb{R} \times \mathbb{R}$, $\text{code} \in \{0, 1\}^*$ indicates the location of the participant and $\text{time} \in \mathbb{N}$ indicates the execution start time of the participant.

If $q_i \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T} \cup \mathcal{U}$, it determines the settings of an existing participant, and if $q_i \notin \mathcal{X}$, it determines the settings of a new participant.

iv. Upon receiving the $\text{CorruptParties}(Q)$ where $Q = \{q_1, \dots, q_{n'}\}$, the challenger runs:

- For a q_i that $q_i.\text{type} = \text{verifier}$, then $v.\text{Code} = q_i.\text{code}$ and $v.\text{Corr} = \text{true}$ for $v \in \mathcal{V}$.
- For each q_i that $q_i.\text{type} = \text{user}$, sets the users' revocation flag as $u.\text{Rev} = \text{true}$ where $u \in \mathcal{U}$, runs $(\text{gpk}') \leftarrow \text{Revoke}(\text{msk}, \text{gpk}, 1)$, and then updates the group public key $\text{gpk} \leftarrow \text{gpk}'$. This applies only if the DBID scheme provides user revocation.
- If there is a q_i that $q_i.\text{type} = \text{prover}$, then for each member p of the set \mathcal{P} , sets their corruption flag $p.\text{Corr} = \text{true}$. If q_i is not corresponding to an existing prover, then create a new prover p and add it to the prover set \mathcal{P} . Set the attributes of the participant p as follows:

a1. location $p.\text{Loc} = q_i.\text{location}$,

a2. execution code $p.\text{Code} = q_i.\text{code}$,

a3. start time $p.\text{St} = q_i.\text{time}$,

a4. corruption flag $p.\text{Corr} = \text{true}$, and

a5. secret key $p.\text{Key} = u.\text{Key}$.

- For each q_i that $q_i.type = actor$, add a new actor x to the set \mathcal{T} , and assign its attributes as follows:

a1. location $x.Loc = q_i.location$,

a2. execution code $x.Code = q_i.code$,

a3. start time $x.St = q_i.time$, and

a4. corruption flag $x.Corr = true$.

- v. The challenger sends the key of the corrupted provers and the key of revoked user to the adversary, i.e., $p.Key$ for all $p \in \mathcal{P}$ such that $p.Corr = true$ and $u.Key$ for all $u \in \mathcal{U}$ such that $u.Rev = true$.

2. **Run:** Challenger activates all participants $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ at time $x.St$ for execution of $x.Code$.

The game ends when the last participant's code completes its execution.

Using the above game, we define four distinct properties for distance-bounding identification schemes. The winning condition of the above game, varies for each property.

Property 4.2.1 (DBID Completeness) Consider a DBID scheme and a DBID game when $Q = \emptyset$ in the $\text{CorruptParties}(Q)$ query and the set \mathcal{P} is not empty.

The DBID scheme is (τ, δ) -complete for $0 \leq \tau, \delta \leq 1$, if the verifier returns $Out_V = 1$ with probability at least $1 - \delta$, under the following assumptions:

- the fast challenge-response rounds are independently affected by noise and at least τ portion of them are noiseless, and
- $\tau > 1 - p_{noise} - \epsilon$ for some constant $\epsilon > 0$.

A complete scheme must have negligible δ to be able to function in the presence of communication noises.

Property 4.2.2 (DBID Soundness) Consider a DBID scheme and a DBID game with the following restrictions:

- \mathcal{P} is nonempty and $\forall p \in \mathcal{P}, v \in \mathcal{V} : d(p.Loc, v.Loc) > \text{DBID}.\mathcal{B}$, and
- in the $\text{CorruptParties}(Q)$ query, $q_i.type \in \{\text{actor}, \text{user}\}$ for all $q_i \in Q$.

In this game the verifier and provers are honest, while the adversary \mathcal{A} corrupts a set of actors and sets their locations (and, if applicable) revokes some users. The corrupted actors are controlled by the adversary, and can simultaneously communicate with multiple provers and the verifier. They can receive a message m from a prover and send m' to the verifier, and vice versa. The certificate of the revoked users are sent to the adversary.

The DBID scheme is γ -sound if the probability of the verifier outputting $\text{Out}_V = 1$ is at most γ .

Lemma 3 A sound scheme according to Property 4.2.2 is resistant against relay attack [BC94], mafia-fraud (Attack 3.1.2), impersonation attack (Attack 3.1.5), and strong-impersonation [ASN17a].

Proof 3 Here we show each of these attacks, separately:

- relay attack [BC94] where the MiM attacker only relays the messages between the honest verifier and a far-away honest prover. The MiM attacker tries to convince the verifier that the prover is located close to the verifier. This attack is achieved by adding extra restrictions on the adversary of Property 4.2.2 as follows:
 - $\forall q_i \in Q$ we have $q_i.code = \text{"relay messages"}$.
- mafia-fraud (Attack 3.1.2) is when there is an honest verifier, an honest far-away prover, and a close-by MiM attacker who tries to convince the verifier that the prover is located close to the verifier. The attacker listens to the legitimate communications for a while, before running the attack as the learning phase. This attack corresponds to adding extra restrictions on the adversary in Property 4.2.2 as follows:

- \mathcal{P} is nonempty, and
 - $\forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \text{DBID}.\mathcal{B}$ for $v \in \mathcal{V}$.
- *impersonation attack (Attack 3.1.5) happens when there is an honest verifier and a single close-by attacker who tries to convince the verifier that the prover is located close to the verifier. The attacker can have a learning phase before running the attack. We can achieve this attack by adding extra restrictions on the adversary of Property 4.2.2 as follows:*
 - \mathcal{P} is nonempty, and
 - $\forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \text{DBID}.\mathcal{B}$ for $v \in \mathcal{V}$, and
 - among all the successful DBID. Π protocols (Π^{succ} set) during the game, $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = \text{fshTime}(\pi), t \notin [p.St, p.St + \text{exLen}(p.Code)]$.
 - *strong-impersonation [ASN17a] happens when either mafia-fraud or impersonation happens. We can achieve this attack by adding extra restrictions on the adversary of Property 4.2.2 as follows:*
 - \mathcal{P} is nonempty, and
 - $\forall q_i \in Q$ we have $d(q_i.location, v.Loc) \leq \text{DBID}.\mathcal{B}$ for $v \in \mathcal{V}$, and
 - among all the successful DBID. Π protocols (Π^{succ} set) during the game, at least one of the following conditions hold:
 - (i) $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = \text{fshTime}(\pi), t \notin [p.St, p.St + \text{exLen}(p.Code)]$
 - (ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{succ}, v \in \mathcal{V} : t = \text{fshTime}(\pi), t \in [p.St, p.St + \text{exLen}(p.Code)]$
 $\wedge d(p.Loc, v.Loc) > \text{DBID}.\mathcal{B}$.

■

We consider two types of attacks by a dishonest prover: far-away dishonest provers (Property 4.2.3), and far-away dishonest provers with a close-by helper (Property 4.2.4).

Property 4.2.3 (DBID Distance-Fraud) Consider a DBID scheme and a DBID game with the following restrictions:

- \mathcal{P} is nonempty and $\forall p \in \mathcal{P}, v \in \mathcal{V} : d(p.Loc, v.Loc) > \text{DBID}.\mathcal{B}$, and
- in the $\text{CorruptParties}(Q)$ query, $q_i.type = \text{prover}$ and $d(q_i.location, v.Loc) > \text{DBID}.\mathcal{B}$ for all $q_i \in Q$ and $v \in \mathcal{V}$.

The DBID scheme is α -DF-resistant if, for any DBID. Π protocol in such game, we have $\Pr[\text{Out}_V = 1] \leq \alpha$.

In the following we define the TF-resistance of DBID protocols.

Property 4.2.4 (DBID Terrorist-Fraud) Consider a DBID scheme and a DBID game with the following restrictions:

- \mathcal{P} is nonempty and $\forall p \in \mathcal{P}, v \in \mathcal{V} : d(p.Loc, v.Loc) > \text{DBID}.\mathcal{B}$, and
- in the $\text{CorruptParties}(Q)$ query, $q_i.type \in \{\text{prover}, \text{actor}\}$ and $d(q_i.location, v.Loc) > \text{DBID}.\mathcal{B}$ for all $q_i \in Q$ that $q_i.type = \text{prover}$ and $v \in \mathcal{V}$.

The DBID scheme is μ -TF-resistant, if the following holds about the above game:

- If the verifier returns $\text{Out}_V = 1$ in the Π protocol of game Γ with non-negligible probability κ , then there is an impersonation attack as a DBID game Γ' with honest verifier, no prover and one close-by actor that takes the view of close-by participants (View_S^Γ) as input, and makes the verifier return $\text{Out}_V = 1$ with probability at least $\kappa - \mu$ in the Π protocol of Γ' game.

Note that this is a formal definition of the terrorist-fraud resistance (Attack 3.1.3) that is based on the recent definitions (such as [Vau13]) and is different from the definition of TF in [Des88], which is the original version of this work. This change in the definition of TF is necessary because here we consider directional antennas. With this new capability, a malicious prover can use directional communication with the verifier and the helper, such that although the TF succeeds, the leaked information does not allow a response generator to be constructed. Using the original approach,

and removing contribution of the verifier's view, allow us to define TF security.

In Lemma 4 we show that if a DBID scheme is TF-resistant (Property 4.2.4), using a directional antenna (as in Figure 4.1) will not affect its security. We only provide an informal proof because a formal proof needs formalizing properties of directional antennas.

Lemma 4 *If a DBID scheme is TF-resistant (Property 4.2.4), it is directional TF-resistant.*

Proof 4 *The main observation is that in a TF attack (Property 4.2.4), all close-by participants, except the verifier, are controlled by the adversary. So, using a directional antenna to communicate with close-by participants such that the verifier is excluded, adds the transmitted message to the view of adversary, and replacing the directional antenna with an omni directional one, does not change this view.*

The messages that are sent to the verifier using directional antenna, will not be included in the impersonation adversary view, i.e., View_S^Γ .

Using property 4.2.4, if there is a successful TF attack against a DBID scheme, the TF-resistant property guarantees existence of an impersonation attacker with non-negligible probability that takes the View_S^Γ as input. Since the view of actors in a directional TF attack will include this view, therefore, in a TF-resistant DBID scheme, having a successful directional TF attack implies future impersonation attack. ■

In fundamental lemma, we relate (i) the local timing of a received message at the verifier, (ii) physical distances traveled by the message, and (iii) the message content. It shows that any response r for the challenge c , that is received by the verifier \mathcal{V} , can be split into two parts r_S and $r_{\mathcal{F}}$ based on the distance of the sender, and each part can be computed from two separate inputs; (a) r_S from the challenge c and the views of close-by participants before seeing c , and (b) $r_{\mathcal{F}}$ from the views of far-away participants before seeing c .

The difference between our fundamental lemma compared to the proposed fundamental lemma of

[Vau14] is twofolds: we allow a received message to be combination of multiple sent messages which is more realistic in wireless communications, and we do not require a global clock.

Lemma 5 (Fundamental Lemma) *Consider a multi-party protocol execution Γ with a distinguished participant $v \in \mathcal{V}$ that measures the local time of events, \mathcal{F} and \mathcal{S} denote the set of far-away and close-by participants, respectively. At local time t , v broadcasts a random message c and waits for a response r . Acc denotes the event that r was received by v at a local time $t' \leq t + \frac{2\mathcal{B}}{\mathcal{C}}$. The message from a participant x is independent from c , if it is the result of running the participant algorithm with view of x , just before seeing c , i.e., $\text{View}_x^\Gamma(\bar{c})$.*

If Acc occurs, then r consists of up to two components $(r_{\mathcal{F}}, r_{\mathcal{S}})$ (i.e., $r = \text{sum}(r_{\mathcal{F}}, r_{\mathcal{S}})$ for a deterministic accumulation function $\text{accumulation} : \mathbb{M}^ \rightarrow \mathbb{M}$, where \mathbb{M} is the set of all possible messages that v may receive), $r_{\mathcal{F}}$ is sent from members of \mathcal{F} and $r_{\mathcal{S}}$ is sent from members of \mathcal{S} , then the following holds:*

$r_{\mathcal{F}} = \text{Msg}_{\mathcal{F} \rightarrow v(t')}(\bar{c})$ and there exist algorithm \mathcal{J}' such that $r_{\mathcal{S}} = \mathcal{J}'(\text{View}_{\mathcal{S}}^\Gamma(\bar{c}), c, \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\bar{c}))$, where $\text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\bar{c})$ is all messages from any member of \mathcal{F} to any member of \mathcal{S} that are independent of c , and $\text{Msg}_{\mathcal{F} \rightarrow v(t')}(\bar{c})$ is the accumulation of all messages from members of \mathcal{F} that get received by the verifier at time t' , which are independent of c . Note that $\text{Msg}_{\mathcal{F} \rightarrow v(t')}(\bar{c}) \in \mathbb{M}$.

Proof 5 *We consider three cases for possible sources of the response r that is received by v ; (i) r is solely sent by a subset $F \subseteq \mathcal{F}$, (ii) r is solely sent by a subset $C \subseteq \mathcal{S}$, or (iii) otherwise. In this proof, we show that the response can be generated according to the lemma in all cases.*

Case (i) *In the first case, let's consider the received message $r = r_{\mathcal{F}}$ by the verifier, as the accumulation¹ of multiple messages (r_1, \dots, r_l) that are sent by participants $F = \{F_1, \dots, F_l\}$ at time $\{t'_1, \dots, t'_l\}$, respectively. All these messages arrive at the verifier at the same time, so the sending times are according to the distance between the sender and the verifier. Without loss of generality,*

¹The accumulation of multiple messages that are received at the same time by the same antenna, depends on the physical properties of the system. We assume it is a deterministic function.

we assume that for $i = \{1 \dots l - 1\}$, the participant F_i is closer to v than participant F_{i+1} , and so $t'_i \geq t'_{i+1}$.

If **Acc** occurs, we have $t'_1 \leq t + 2\mathcal{B} - \frac{d(v, F_1)}{c}$, and since $d(v, F_1) > \mathcal{B}$, then we have $t'_1 < t + \frac{d(v, F_1)}{c}$ which is before the time F_1 could see the challenge c . We have the same inequality for other participants, so $r = r_{\mathcal{F}} = \text{Msg}_{\mathcal{F} \rightarrow v(r')}(\bar{c}) = \text{sum}(r_1, \dots, r_l)$ is independent of c .

Case (ii) In the second case, consider the received message $r = r_{\mathcal{S}}$ by the verifier, as the accumulation of multiple messages (r_1, \dots, r_l) that are sent by participants $C = \{C_1, \dots, C_l\}$ at time $\{t'_1, \dots, t'_l\}$, respectively. All these messages arrive at the verifier at the same time, so the sending times are according to the distance between the sender and the verifier. Without loss of generality, we assume that for $i = \{1 \dots l - 1\}$, the participant F_i is closer to v than participant F_{i+1} , and so $t'_i \geq t'_{i+1}$.

We construct an algorithm \mathcal{J}' that simulates all close-by participants $x \in \mathcal{S}$ in the time range $[t + \frac{d(v, x)}{c}, t + \frac{2\mathcal{B} - d(v, x)}{c}]$ (i.e., from the event of seeing c , until before it's too late to send a message to the verifier and make **Acc** occur). The simulation is in parallel and in chronological order. The output of \mathcal{J}' is the message $r = r_{\mathcal{S}}$ delivered to v as the accumulation of messages sent from $C \subseteq \mathcal{S}$ at time $t' \leq t + \frac{2\mathcal{B} - d(v, C_1)}{c}$, where C_1 is the closest responder.

Here we claim that the input (m) of each responder $x \in C$ is either part of the input of \mathcal{J}' in the lemma. In order to prove it, we consider four cases about the source of m :

- m comes from the internal view of x before seeing c , i.e., $\text{View}_x^\Gamma(\bar{c})$; since $x \in \mathcal{S}$, then $\text{View}_x^\Gamma(\bar{c}) \in \text{View}_{\mathcal{S}}^\Gamma(\bar{c})$ that is included in the input of the simulator.
- m comes from far-away participants; in this case it must be independent from c due to the distance constraints, and so $m \in \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\bar{c})$, that is included in simulator's input.
- m comes from v ; since v only sends c before r , then the message m can be either c ,

or already is in the view of the close-by participants before c , i.e., $\text{View}_S^{\Gamma}(\bar{c})$.

- m comes from a close-by participant $y \in \mathcal{S}$; then the above three cases about x , applies to y too. This part is a recursive argument till there is no close-by participant left to send the message.

Case (iii) In the third case, consider the received message r by the verifier, as the accumulation of messages sent from close-by participants (r_S) and messages sent from far-away participants ($r_{\mathcal{F}}$). Note that all messages are delivered at the same time t' to v . We have thus showed that there are algorithms that can generate r_S and $r_{\mathcal{F}}$ separately. Therefore by applying the accumulation function $r = \text{sum}(r_S, r_{\mathcal{F}})$, the algorithm can compute the response message r with correct timing.

■

4.3 DBID Construction: POXY

In this section we present a new DBID construction as an extension of DBPK-log⁺ [ASN14] and DBPK-log [BB04]. This protocol uses Pedersen [Ped92] cryptosystem (See Algorithm 2.6.1). As a DBID ($\mathbb{X}; \mathbb{Y}; \mathbb{S}; \text{Init}; \text{KeyGen}; \Pi; \mathcal{B}; p_{\text{noise}}$) protocol, Poxy consists of all operations in DBID scheme as follows;

4.3.1 $(msk, gpk) \leftarrow \text{Init}(1^\lambda)$

The group manager initializes a Pedersen commitment with λ bit security: chooses a large λ -bit strong prime p , such that $p = 2q + 1$ for a large prime q . It also chooses the group generator g for \mathbb{Z}_p^* and a random element $h \in_R \mathbb{Z}_p^*$.

The group manager initiates a certificate mechanism for validating the public key of the user, i.e., creates a certificate key pair $(sk^{\text{Cert}}, pk^{\text{Cert}})$. We omit this mechanism from the rest of this section for simplicity. So we have $msk = (sk^{\text{Cert}})$ and $gpk = (p, q, g, h, pk^{\text{Cert}}, \Xi)$ where $\Xi = \emptyset$.

4.3.2 $(sk, pk) \leftarrow \text{KeyGen}(msk, gpk)$

Assume $l - 1$ users have joined the group and their public keys are in the set $\Xi = \{pk_1, \dots, pk_{l-1}\}$ that is published by the group manager. For the l^{th} user, the group manager generates a key pair (sk, pk) , such that $sk \in_R \mathbb{Z}_{p-1}$ and $pk = \text{Commit}(sk; 0) = g^{sk} \pmod{p}$, where $\text{Commit}(u; v)$ is Pedersen commitment ($= g^u h^v \pmod{p}$). The group manager securely sends the key pair to the new user and adds the public key pk to the set Ξ .

4.3.3 $accept/reject \leftarrow \Pi\{P(sk, pk) \leftrightarrow V(pk)\}$

When a prover of a registered user wants to run the DBID. Π protocol with the verifier, they will follow the protocol described in Figure 4.5.

The current form of the protocol is for better readability. However, the actual order of the messages is in the order of Σ^* -protocol, that consists of three type of messages: commitment, challenge and response. In Figure 4.5, the messages are mark by three signs; (A) as commitment, (C) as challenge, and (r) as response of a Σ^* -protocol. If we rearrange the messages (including the messages of ZKP sub-protocols) based on the type, according to Definition 2.4.4, then Poxy. Π becomes a Σ^* -protocol.

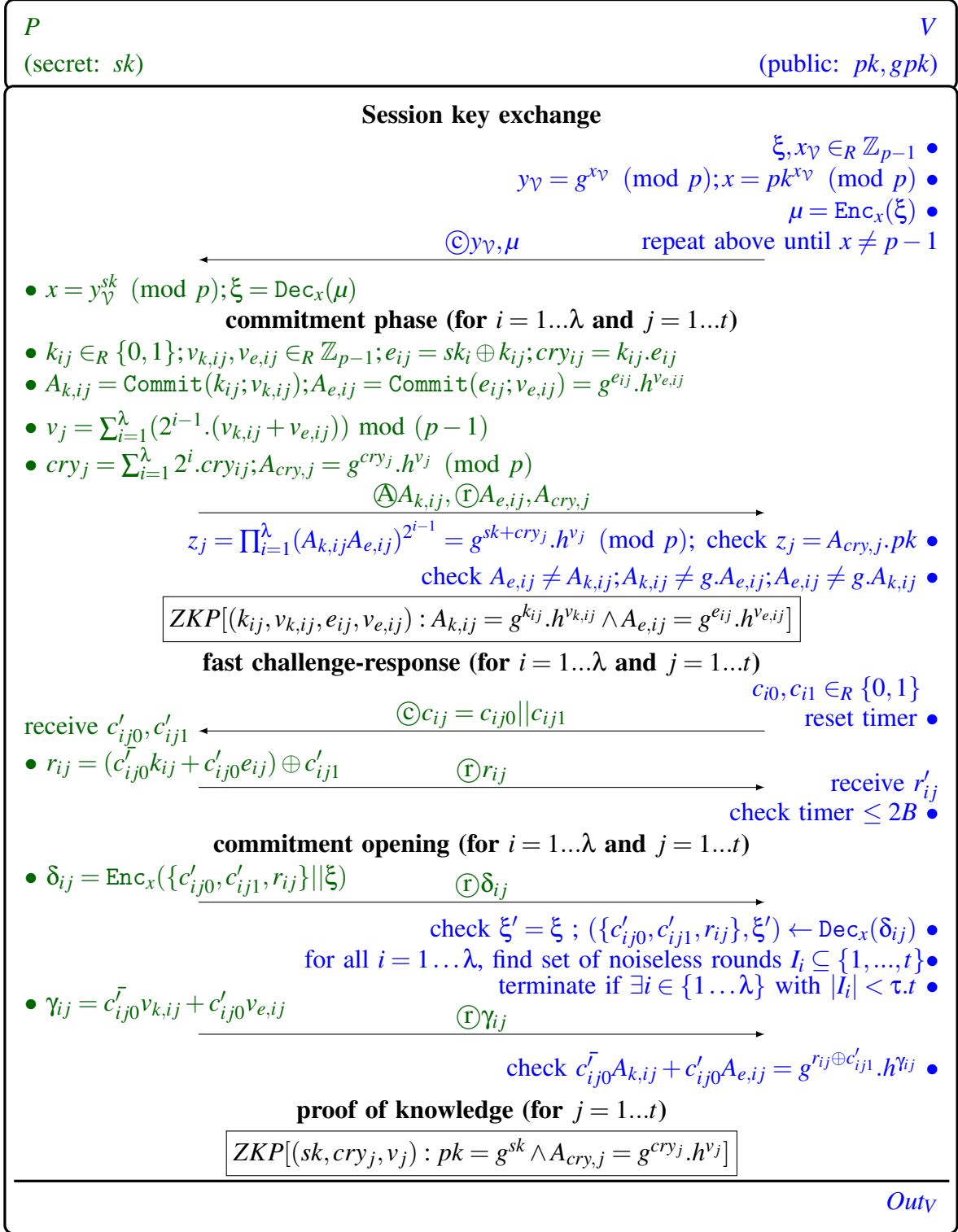


Figure 4.5

4.5 (previous page): Π protocol of Poxy scheme. Commit is Pedersen commitment scheme. (Enc, Dec) is a secure symmetric encryption scheme. ZKP is a zero-knowledge proof-of-knowledge protocol. The notation \textcircled{A} by a message, indicates that the message is considered as commitment of Σ^* -protocol. The notations \textcircled{C} and \textcircled{R} indicate the challenge and response messages of Σ^* -protocol, respectively. cry_j is the vector of carry-ons in addition of k_j and e_j .

4.3.4 $(\text{msk}', \text{gpk}') \leftarrow \text{Revoke}(\text{msk}, \text{gpk}, i)$

The group manager removes the i^{th} public key from the set Ξ . i.e., $\Xi := \Xi \setminus \{pk_i\}$.

4.3.5 Security Analysis

In this section we provide the security analysis of Poxy protocol.

Theorem 1 *Assuming ZKP is a κ -sound (Definition 2.4.1) and ζ -zero-knowledge identification protocol (Definition 2.5.4) for negligible values of κ and ζ , and (Enc, Dec) is an IND-CCA symmetric encryption scheme;*

Poxy is (τ, δ) -complete, μ -TF-resistant, γ -sound, α -DF-resistant and also zero-knowledge (Definition 2.5.4) DBID scheme for negligible values of δ , μ , γ and α , when t is linear in security parameter λ , and $\lambda \cdot (1 - p_{\text{noise}} - \epsilon) > \lambda \cdot \tau \geq \lambda - (\frac{1}{2} - 2\epsilon) \lceil \frac{\lambda}{2} \rceil$ for some constant $\epsilon > 0$.

Lemma 6 (Completeness) *By assuming $1 - p_{\text{noise}} - \epsilon > \tau$ for some constant $\epsilon > 0$, and (Enc, Dec) is an IND-CCA symmetric encryption scheme;*

Poxy is a (τ, δ) -complete DBID protocol for some negligible value of δ .

Proof 6 *Consider a DBID game with Poxy scheme, in which there is no actor, and the provers and the verifier are honest, i.e., $\forall x \in \mathcal{P} : x.\text{Code} = \text{Poxy}.\Pi.P(.) \wedge d(x.\text{Loc}, v.\text{Loc}) \leq \mathcal{B}$ for $v \in \mathcal{V} : v.\text{Code} = \text{Poxy}.\Pi.V(.)$. The verifier has access to the correct public key of provers. In this proof, we calculate the success chance of an honest prover in a Π protocol.*

let's assume the verifier sends the challenge sequence $[c] = ([a], [b])$, where $[a] = [a^1 \dots a^\lambda]$, and $[b] = [b_1 \dots b_m]$ for $m \geq 0$. The prover receives $[c'] = ([a'], [b'])$ such that $\forall i \in \{1, \dots, \lambda\}, j \in \{1, \dots, t\} : \Pr[a_{ij} = a'_{ij}] = 1 - p_{\text{noise}}$. Correspondingly, the prover sends the response sequence $[r] = ([d], [e])$ and the verifier receives $[r'] = ([d'], [e'])$, where $\forall j \in \{1, \dots, \lambda\}, \forall j \in \{1, \dots, t\} : \Pr[d_{ij} = d'_{ij}] = 1 - p_{\text{noise}}$.

After the commitment opening phase, the verifier is able to find the noisy rounds (except with probability $\text{Adv}_{\text{Corr}}^{\text{Enc}}$ that is negligible). The probability of having at least τ noiseless fast challenge-response rounds for each $j \in \{1, \dots, t\}$ is $\text{Tail}(\lambda, \tau, \lambda, 1 - p_{\text{noise}})$ (Tail function is defined in Theorem 1). As a result, the failure chance of the protocol is t times $1 - \text{Tail}(\lambda, \tau, \lambda, 1 - p_{\text{noise}})$, which is less than $e^{-2\varepsilon^2\lambda}$ based on Chernoff bound (Theorem 1). And we have $t \cdot e^{-2\varepsilon^2\lambda} < \text{negl}(\lambda)$ if t is linear to λ . ■

Lemma 7 (Distance-Bounding) By assuming $\tau \cdot \lambda \geq \lambda - (\frac{1}{2} - \varepsilon) \lfloor \frac{\lambda}{2} \rfloor$ for some constant $\varepsilon > 0$, Diffie-Hellman key exchange is computationally unforgeable, ZKP is a κ -sound (Definition 2.4.1) and ζ -zero-knowledge identification (Definition 2.5.4) for negligible κ and ζ :

Poxy is an α -DF-resistant DBID protocol for negligible value of α .

Proof 7 Consider a DBID game of Pox scheme with no actors (i.e., $\mathcal{T} = \emptyset$), honest verifier (i.e., $v \in \mathcal{V} : v.\text{Code} = \text{Pox}.\Pi.V(.)$) and far-away corrupted provers (i.e., $\forall x \in \mathcal{P} : d(x.\text{Loc}, v.\text{Loc}) > \mathcal{B} \wedge x.\text{Code} \neq \text{Pox}.\Pi.P(.)$) that might overlap in their execution time (i.e., $x, y \in \mathcal{P} : y.\text{St} < x.\text{St} + \text{exLen}(x.\text{Code}) \leq y.\text{St} + \text{exLen}(y.\text{Code})$).

In a successful Π protocol, the verifier gets $\xi' = \xi$ at the end of commitment opening phase, which implies that the prover has the correct value of x , unless negligible probability $\text{Adv}_{\text{forge}}^{\text{DH}}$, as the forgery chance of the semi-fresh Diffie-Hellman key exchange protocol [DH76], which is used in session key exchange phase. If the adversary succeeds in commitment and proof-of-knowledge phases, then they know a certain $\{(e'_j, k'_j)\}_{j=1 \dots t}$ that satisfies " $e'_j \oplus k'_j = sk$ " for all $j \in \{1, \dots, t\}$. Therefore, they can efficiently find sk , unless negligible probability $(2\lambda + t)\text{Adv}_{\text{sound}}^{\text{ZKP}} + \text{Adv}_{\text{forge}}^{\text{DH}} =$

$$(2\lambda + t)\kappa + \text{Adv}_{\text{forge}}^{DH}.$$

Since the value of sk is chosen randomly, then we have $\Pr[e_{ij} = k_{ij}] = \frac{1}{2}$ for $i = 1 \dots \lambda$ and $j = 1 \dots t$. In a DF attack, any collaboration of far-away provers in sending the response r_i is independent from the challenge bit c_i , according to the fundamental lemma (Lemma 5). Therefore, for the cases that $e_{ij} \neq k_{ij}$ (i.e., half of the rounds), the success chance of adversary in sending the correct r_{ij} is $\frac{1}{2}$. As a result, the success chance of adversary in guessing the correct responses is limited by $\text{Tail}(\lfloor \frac{\lambda}{2} \rfloor, \tau \cdot \lambda - \lceil \frac{\lambda}{2} \rceil, \frac{1}{2})^\lambda$, which is negligible based on Chernoff bound (Lemm 1). ■

We prove TF-resistance of Poxy in Lemma 9 that uses the following lemma.

Lemma 8 (Extractor) Consider a DBID game Γ of a TF attack (Property 4.2.4), for Poxy scheme. If there is a Π protocol in the game Γ in which, the verifier returns $\text{Out}_V = 1$ with non-negligible probability p , then there is a PPT extractor \mathcal{E} , that takes the view of all close-by participants, except the verifier (View_S^Γ) as input, and outputs $sk' = sk$ with probability $p - \mu$ for negligible value of μ . This holds assuming that ZKP is κ -sound (Definition 2.5.4).

Proof 8 (Extractor) Let's assume there is a TF adversary \mathcal{A} that succeeds in Π protocol with non-negligible probability p , i.e., generates a transcript $\xi = (A, [c], [r])$ that is accepted by the verifier with probability p . We construct a PPT extractor algorithm \mathcal{E} for the secret key.

In a Π protocol from game Γ , the sequence of all challenges $[c]$ (slow and fast) is chosen randomly and broadcasted by the honest verifier. We define $[r] = [r]^{fast} || [r]^{slow}$ and $[c] = [c]^{fast} || [c]^{slow}$ where the superscripts show the type of the phase of the challenges.

Let S be the event that for all $i = \{1 \dots \lambda\}$, and $j \in I_i$, the verifier's check $r_{ij} \oplus c_{ij1} = c_{ij0}k_{ij} + c_{ij0}(sk_i \oplus k_{ij})$ hold true, where $c_i = c_{i0} || c_{i1}$. This can be verified by checking success of all ZKP's in commitment phase, all the ZKP's in PoK phase and all the checks in commitment opening phase. In other words, when in commitment opening phase all checks succeed, we have $r_{ij} \oplus c_{ij1} = c_{ij0}k_{ij} + c_{ij0}(e_{ij})$. And when all ZKP's of commitment phase succeed, we have the commitment to every bit of k_{ij} and e_{ij} for $i = \{1 \dots \lambda\}$ and $j = \{1 \dots t\}$, that builds z_j as the commitment to $e_j \oplus k_j$.

And when all ZKP's of commitment phase succeed, we have $sk = e_j \oplus k_j$ for all $j = \{1 \dots t\}$. This implies the occurrence of S .

Since ZKP is κ -sound, we conclude that at least for one ZKP we have $\Pr[\text{succ ZKP} | \neg S] \leq \kappa$ and then $\Pr[\text{succ ZKP}, \neg S] \leq \kappa$, where $\neg S$ indicates negation of S . So we have $\Pr[\text{valid } \xi, \neg S] \leq \kappa$.

Based on the fundamental lemma (Lemma 5), any valid response r_{ij} that is received by the verifier consists of two parts $r_{\mathcal{F}}$ and $r_{\mathcal{S}}$ that $r_{\mathcal{F}} = \text{Msg}_{\mathcal{F} \rightarrow v(t')}(c_{ij})$ and there exists algorithm $\mathcal{J}' : r_{\mathcal{S}} = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(c_{ij}), c_{ij}, \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(c_{ij}))$. We have $r_{ij} = \text{sum}(r_{\mathcal{F}}, r_{\mathcal{S}})$, for a deterministic function $\text{sum}()$ that is determined by the physical communication channel. We assume there exist a deterministic subtraction function $\text{sub}()$ such that for any $x = \text{sum}(z, y)$, we have $z = \text{sub}(x, y)$ and $y = \text{sub}(x, z)$.

We consider the view of close-by participants before sending the response r_{ij} , i.e., $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{ij})$, relative to the view of the close-by participants before seeing the challenge c_{ij} , i.e., $\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{ij})$. In the time period between receiving the challenge c_{ij} and sending r_{ij} , the close-by participants can receive messages from two different sources: the verifier, and the far-away participants. The only message from the verifier in this period is c_{ij} and we indicate the messages from the far-away participants as $\text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{ij})$ which is independent from c_{ij} . So we have $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{ij}) = \text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{ij}) || c_{ij} || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{ij})$.

We conclude that there is an algorithm \mathcal{J}' that takes $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{ij})$ and generates $r_{\mathcal{S}}$, such that the correct response r_{ij} is calculated as $r_{ij} = \text{sum}(\text{Msg}_{\mathcal{F} \rightarrow v(t')}(c_{ij}), r_{\mathcal{S}})$. Note that $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_i)$ includes the challenge c_{ij} .

Since there is an algorithm \mathcal{J}' that generates $\text{resp}_{ij} = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{ij}) || c_{ij} || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{ij}))$, then we construct the algorithm \mathcal{J} that calls \mathcal{J}' four times for different values of c_{ij} with the following inputs: $\text{resp}_{ij}^d = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{ij}) || d || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{ij}))$ for $d = \{00, 01, 10, 11\}$. As a result, \mathcal{J} returns $\{\text{resp}_{ij}^d\}_{d=\{00,01,10,11\}}$ such that the output of the function $\text{sum}(\text{resp}_{ij}^d, r_{\mathcal{F}})$ is the correct answer when the challenge is $c_{ij} = d$, which makes correct resp_{ij}^d . Note that the value of $r_{\mathcal{F}}$ is not known to the extractor.

From the final view of the close-by participants, we can find their partial view at the time of sending response r_{ij} for all $j = 1 \dots \lambda$ and $i = 1 \dots t$, and then call the algorithm \mathcal{J} . So we can calculate $(resp_{ij}^{00}, resp_{ij}^{01}, resp_{ij}^{10}, resp_{ij}^{11})$ for all $i = 1 \dots \lambda$ and $j = 1 \dots t$, from the final view of the close-by participants.

Since we have $sub(sum(resp_{ij}^{11}, r_{\mathcal{F}}); sum(resp_{ij}^{01}, r_{\mathcal{F}})) = sub(resp_{ij}^{11}; resp_{ij}^{01})$, and $sub(sum(resp_{ij}^{10}, r_{\mathcal{F}}); sum(resp_{ij}^{00}, r_{\mathcal{F}})) = sub(resp_{ij}^{10}; resp_{ij}^{00})$, then for all values of $resp_{ij}^{00}$, $resp_{ij}^{01}$, $resp_{ij}^{10}$, $resp_{ij}^{11}$ and $r_{\mathcal{F}}$, calculating the difference between the two correct responses for both values of the challenge bit c_{ij0} given c_{ij1} , is equal to the difference between $resp_{ij}^{1||c_{ij1}}$ and $resp_{ij}^{0||c_{ij1}}$, which are computable from the final view of close-by participants ($View_{\mathcal{S}}^{\Gamma}$). In the following, we use function $sub^{\oplus}(\cdot; \cdot)$ that is defined as: $sub^{\oplus}(a; b) := 0$ if $sub(a; b) = 0$, and $sub^{\oplus}(a; b) := 1$ otherwise.

We build the extractor \mathcal{E} as follows: \mathcal{E} runs $(resp_{ij}^{00}, resp_{ij}^{10}) \leftarrow \mathcal{J}(View_{\mathcal{S}}^{\Gamma})$ for all $i = 1 \dots \lambda$ and $j = 1 \dots t$, and finds $\xi_{ij} = sub^{\oplus}(resp_{ij}^{10}; resp_{ij}^{00})$, and then calculates the key bits $sk'_i = \text{majority}(\xi_{ij})$ over $j = \{1 \dots t\}$.

Note that in a simple case that the received message at the receiver is sent from a single source, we have $sub^{\oplus}(resp_{ij}^{10}; resp_{ij}^{00}) = resp_{ij}^{10} \oplus resp_{ij}^{00}$ and also for all $d = \{0, 1\}$ we have $resp_{ij}^{d||1} = resp_{ij}^{d||0} \oplus 1$. In the following we calculate the success chance of this extractor.

A received response $r_{ij}^{d||b}$ for $d, b \in \{0, 1\}$ is correct if $r_{ij}^{d||b} = (\bar{d}.k_{ij} + d.e_{ij}) \oplus b$. For succeeding in the Π protocol (event S), for at least $\tau.t$ values of $j \in \{1 \dots t\}$ the received response should be correct for all values of $i \in \{1, \dots, \lambda\}$. Since the challenge $c_{i,j}$ is chosen randomly and we assume the secret sk is chosen randomly too, then the distribution of a correct response is uniform. So the messages of far-away provers, i.e., $r_{\mathcal{F}}$, have at most $\frac{1}{2}$ chance of being the correct response. Therefore, for at least $\tau.t$ values of $j = \{1 \dots t\}$, the close-by actors help the prover.

An extracted response $resp_{ij}^{d||0}$ for $d \in \{0, 1\}$ is correct if $sum(resp_{ij}^{d||0}, r_{\mathcal{F}}) = \bar{d}.k_{ij} + d.e_{ij}$. We define $R_{ij} \in \{0, 1, 2\}$ as the number of challenge bits $d \in \{0, 1\}$ for which the extracted response

$\text{resp}_{ij}^{d||0}$ is correct. If we have $R_{ij} = 2$ then $\xi_{ij} = sk_i$, but if $R_{ij} = 1$ then we might have $\xi_{ij} \neq sk_i$.

Consider $R = (R^1, \dots, R^\lambda)$ for vector $R^i = (R_{i1}, \dots, R_{it})$, where R_{ij} is defined as above and calculated by comparing the $(\text{resp}_{ij}^{00}, \text{resp}_{ij}^{10})$ with correct responses for all $i = 1 \dots \lambda$ and $j = 1 \dots t$. We define the set \mathcal{R} as all vectors in $\{0, 1, 2\}^t$ such that at least $\lfloor \frac{t}{2} \rfloor + 1$ values are 2. For a vector $R^i \in \mathcal{R}$ we have $\xi_{ij} = sk_i$.

Since the verifier selects the challenges randomly, then knowing R_{ij} allows us to find the probability that the response $r_{ij} = \text{sum}(\text{resp}_{ij}, r_{\mathcal{F}})$ is correct: if $R_{ij} = 2$ then this probability is 1, otherwise this probability is at most $\frac{1}{2}$ (a response can always be guessed randomly). For any $R^i \notin \mathcal{R}$, at least $\lceil \frac{t}{2} \rceil$ values of $R_{i,j} \neq 2$, and at least $\tau \cdot t - \lfloor \frac{t}{2} \rfloor$ of them have to be guesses randomly for success. Therefore, the probability of success in the i^{th} round by having $R^i \notin \mathcal{R}$ is limited by $p_B = \text{Tail}(\lceil \frac{t}{2} \rceil, \tau \cdot t - \lfloor \frac{t}{2} \rfloor, \frac{1}{2})$, where $\text{Tail}(n, k, \rho) := \sum_{i=k}^n \binom{n}{i} \rho^i (1 - \rho)^{n-i}$.

We define W as the random variable showing the number of i 's that vector $R^i \notin \mathcal{R}$. So we have $\Pr[S|W = w] \leq p_B^w$. So $\Pr[S, W = w] \leq p_B^w \Pr[W = w]$ and then $\Pr[S, W \geq w] \leq p_B^w$. As a result, we have:

$$\Pr[W \geq w, \text{valid } \xi] \leq \Pr[\neg S, \text{valid } \xi] + \Pr[S, W \geq w] \leq \kappa + p_B^w$$

Each index j where $sk_j \neq sk'_j$, corresponds to $R_j \notin \mathcal{R}$. Therefore, having the verifier outputting $\text{Out}_V = 1$ (i.e., ξ is valid) and the extractor giving at least w errors occurs with probability bounded by $\mu = \kappa + p_B^w$. This implies that we can build a key extractor from View_S^Γ that follows the success chance of the TF attack (i.e., p), except with probability μ , which is negligible due to Chernoff bound (Lemm 1). ■

Lemma 9 (TF-resistance) Assuming ZKP is κ -sound (Definition 2.5.4) for negligible κ ;

Poxy is a μ -TF-resistance DBID scheme for negligible value of μ .

Proof 9 Consider a DBID game with single limitation of having honest verifier (i.e., $\mathcal{V}.\text{Code} = \text{ID.V}(\cdot)$). Based on the TF definition (Property 4.2.4), here we show that if there is a TF attacker

that succeeds with probability p , then there exists an impersonator that take the view of close-by actors and succeeds with probability $p - \mu$ for negligible value of μ .

If there is a TF attacker that succeeds with probability p , then based on Lemma 8, there is an extractor that takes the view of close-by actors and returns the secret key of prover $sk' = sk$ with success chance $p - \mu$ for negligible μ .

Therefore, the impersonator first runs the extractor and gets prover's secret key sk' , and then runs the prover protocol $P(sk')$ with the verifier. The success chance of the impersonator is at least $p - \mu$. ■

Lemma 10 (Zero-Knowledge) By assuming ZKP and Enc operations are computationally negligible zero-knowledge; Π protocol of Poxy scheme with honest prover is ζ -zero-knowledge protocol for negligible ζ (Definition 2.5.4).

Proof 10 Here we show that in Π operation, given two participants $P(sk)$ and $V^*(pk, gpk)$, there exists a simulator $S(pk, gpk)$ such that the view of $V^*(pk, gpk)$ in the interaction $V^*(pk, gpk) \leftrightarrow P(sk)$ is computationally indistinguishable from the output of $S(pk, gpk)$. We basically show that there is a simulator that consist of some smaller simulators for each part of the protocol.

The view of adversary at the end is as follows: $\forall j \in \{1, \dots, t\}, i \in \{1, \dots, \lambda\}, b \in \{''k'', ''e''\} : (\text{View}_V(\text{ZKP})^{t, \lambda+t}, x_V, y_V, \mu, \xi, A_{b,ij}, r_{ij}, c_{ij}, \delta_{ij}, \gamma_{ij})$ for $\forall j \in \{1, \dots, t\}, i \in \{1, \dots, \lambda\}$, where the adversary chooses values $x_V, y_V, \xi, \mu, \{c_{ij}\}$.

Since ZKP and Enc operations are assumed to be ζ -zero-knowledge for negligible ζ (Definition 2.5.4), then there exists a simulator for the view of V^* after these operations, that takes the view of V^* as input and returns an output indistinguishable from the normal case. By this means, we can remove the prover side of ZKP operations. In the Diffie-Helman key exchange session, the verifier generates the values of $(x_V, y_V, \mu, \xi, \delta_{ij})$. Therefore, we can remove them from the view of adversary, since it can re-generate them later. Therefore, view of adversary is reduced to: $\forall j \in \{1, \dots, t\}, i \in \{1, \dots, \lambda\} : (A_{k,ij}, A_{e,ij}, r_{ij}, \gamma_{ij}, c_{ij})$, where the adversary chooses the values $\{c_{ij}\}$.

Since the relation between e and k is no longer checked by the above removals, then we can make their mutual relation to be random on the prover side. i.e., choose e randomly. Therefore both values of e and k will be random and the view of verifier in this case is indistinguishable from the case when e is computed as $e_{ij} = sk_i \oplus k_{ij}$ for $j \in \{1, \dots, t\}, i \in \{1, \dots, \lambda\}$. In this modified case, the value of sk is no longer used. Therefore, we can replace sk at the prover with randomness and still be indistinguishable at the verifier. As a result, we built a simulator that doesn't use the secret of prover (sk) and produces indistinguishable output from the output of verifier while it's interacting with real prover. ■

Lemma 11 (Soundness) By assuming t is polynomial in λ , $\tau > \frac{1}{2} + \epsilon$ for some constant $\epsilon > 0$, ZKP is a κ -sound (Definition 2.4.1) and ζ -zero-knowledge authentication (Definition 2.5.4) for negligible κ and ζ in λ , and Enc is ζ -zero-knowledge;

Poxy is a γ -sound DBID (Property 4.2.2) protocol for negligible γ .

Proof 11 According to the DBID game settings, we have some lists of provers $\mathcal{P}^j \in \mathcal{P}$ that there is no overlap in the execution time of any list \mathcal{P}^j , however the provers of two different lists $\mathcal{P}^j \neq \mathcal{P}^i$ can run simultaneously. The corrupted actors \mathcal{T} are controlled by the adversary. In this game, the adversary succeeds if any of the following is true;

- (i) $\exists t \in \mathbb{N}, \forall p \in \mathcal{P} : t = \text{Time}(\text{Out}_\gamma = 1), t \notin [p.\text{St}, p.\text{St} + \text{exLen}(p.\text{Code})]$, or
- (ii) $\exists (p, t) \in (\mathcal{P}, \mathbb{N}) : t = \text{Time}(\text{Out}_\gamma = 1), t \in [p.\text{St}, p.\text{St} + \text{exLen}(p.\text{Code})], d(p, \mathcal{V}) > \mathcal{B}$

In this proof, we calculate the success chance of the adversary in both conditions.

Here we introduce two time periods: the learning phase is from the beginning of the game till the beginning of the first session that makes the adversary to win, and the attack session, that is right after the learning phase till end of the ID operation that makes the adversary to win.

Let's assume the adversary uses $\{A_{e,ij}\}$ and $\{A_{k,ij}\}$ in the commitment phase of the attack session, which are the bit commitments of e and k . Let's assume the response table used in fast phase is e'

and k' , which is known by the close-by participant.

In the first condition (i.e., no active prover during attack session), since there is ZKP for all $2\lambda t$ of the committed values ($\{A_{e,ij}\}$ and $\{A_{k,ij}\}$), the adversary should know the committed values e and k , unless with negligible probability $2\lambda.t.\text{Adv}_{\text{sound}}^{\text{ZKP}} = 2\lambda.t.\kappa$. If e and k are correct (i.e., $e_{ij} = sk_i \oplus k_{ij}$ for all $j \in \{1, \dots, t\}, i \in \{1, \dots, \lambda\}$), then the adversary can efficiently calculate the value of sk , which is in contradiction with Lemm 10 unless with probability $\zeta = \text{Adv}_{\text{ZKP}}$. If e and k are incorrect, then the adversary would not be able to pass the ZKP of proof-of-knowledge phase, unless with negligible probability $t.\text{Adv}_{\text{sound}}^{\text{ZKP}} = t.\kappa$.

In the second condition (all active provers are far-away from verifier during attack session), based on Lemm 10, we know that the close-by participants (actors) have negligible information about sk . In this case, for each row of the response table (i.e., the two bits e'_{ij} and k'_{ij}) there is one bit uncertainty ($\Pr[e_{ij} = e'_{ij}, k_{ij} = k'_{ij}] = \frac{1}{2}$). Otherwise, the adversary would gain some information about sk_i . Therefore, by assuming that the challenge bits of fast phase are chosen randomly, then for half of the fast rounds, the verifier sends the challenge value that the actor needs to make a guess for the correct answer. As a result, for each value of $i \in \{1, \dots, \lambda\}$ the success chance of related round in fast phase is $\frac{1}{2}^{\frac{t}{2} - (1-\tau)t} = \frac{1}{2}^{(\tau - \frac{1}{2})t}$. Therefore, the success chance of the adversary is limited by $2\lambda.t.\kappa + t.\kappa + \frac{1}{2}^{(\tau - \frac{1}{2})t.\lambda} + \zeta$, which is negligible. ■

4.4 DBID Construction: ProProx [Vau14]

ProProx scheme is a public key DB protocol [Vau14] that fits our DBID model. We also prove security of the protocol in this model. The details of the operations of ProProx is given below. Let λ and n be the security parameters that are linearly related.

4.4.1 $(msk, gpk) \leftarrow \text{Init}(1^\lambda)$

The group manager initializes a Goldwasser-Micali cryptosystem [GM84] with λ bit security: chooses $N = p \cdot q$ and chooses θ that is a quadratic residue modulo N . It also chooses $b \in \{0, 1\}^n$ with Hamming weight of $\lfloor \frac{n}{2} \rfloor$. The group master key is $msk = (p, q)$; the group public key is $gpk = (N, b, \theta, \Xi)$ where $\Xi = \emptyset$.

4.4.2 $(sk, pk) \leftarrow \text{KeyGen}(msk, gpk)$

Assume $l - 1$ users have joined the group and their public keys are in the set $\Xi = \{pk_1, \dots, pk_{l-1}\}$ that is published by the group manager. For the l^{th} user, the group manager generates a key pair (sk, pk) , such that $sk \in_R \{0, 1\}^\lambda$ and pk is the output of a homomorphic and deterministic commitment scheme $Com_H()$ on $sk = (sk_1 \dots sk_\lambda)$; that is $pk = Com_H(sk) = (Com(sk_1; H(sk, 1)), \dots, Com(sk_\lambda; H(sk, \lambda)))$, where $Com(u; v)$ is Goldwasser-Micali encryption ($= \theta^u v^2 \pmod{N}$) and H is a one-way hash function. The group manager securely sends the key pair to the new user and adds the public key pk to the set Ξ .

4.4.3 $accept/reject \leftarrow \Pi\{P(sk, pk) \leftrightarrow V(pk)\}$

When a prover (\mathcal{P}_l) of a registered user wants to run the DBID. Π protocol with the verifier, they will follow the protocol described in Figure 3.12.

4.4.4 $(msk', gpk') \leftarrow \text{Revoke}(msk, gpk, i)$

The group manager removes the i^{th} public key from the set Ξ . i.e., $\Xi := \Xi \setminus \{pk_i\}$.

4.4.5 Security Analysis

To prove that the above protocol satisfies our security definition, we first note that the Π protocol of ProProx scheme (*i.e.*, Figure 3.12) can be seen as a Σ^* -protocol (Definition 2.4.4). This is true because, assuming the agreement step (on the value of I) and the ZKP step can be in the form of Σ^* -protocols, therefore running them after each other is also a Σ^* -protocol. This is because we can consider all first message commitment of the protocols as a single commitment phase, and all verification functions stay at the end. The remaining challenge and response messages are run sequentially and form the challenge and responses of the combined protocol.

Theorem 2 *Assuming $\text{Com}(u;v)$ is a perfect binding and computationally hiding homomorphic bit commitment scheme (Definition 2.6.3), $\text{Com}_H()$ is one-way function (Definition 2.5.5), and ZKP is a κ -sound (Definition 2.4.1) and ζ -zero-knowledge identification protocol (Definition 2.5.4) for negligible values of κ and ζ :*

ProProx is (τ, δ) -complete, μ -TF-resistant, γ -sound and α -DF-resistant DBID scheme for negligible values of δ, μ, γ and α , when n is linear in security parameter λ , and $n \cdot (1 - p_{\text{noise}} - \epsilon) > n \cdot \tau \geq n - (\frac{1}{2} - 2\epsilon) \lceil \frac{n}{2} \rceil$ for some constant $\epsilon > 0$.

ProProx is proven to be *complete*, *DF-resistant* and *zero-knowledge* (Definition 2.5.4) in [Vau14]. Our definitions of these properties remain unchanged. So we only need to prove *TF-resistance* and *soundness* properties of ProProx scheme.

We prove TF-resistance of ProProx in Lemma 13 that uses the following lemma.

Lemma 12 (Extractor) *Consider a DBID game Γ with TF attack (Property 4.2.4), for ProProx scheme. If there is a Π protocol in the game Γ in which, the verifier returns $\text{Out}_V = 1$ with non-negligible probability p , then there is a PPT extractor \mathcal{E} , that takes the view of all close-by participants, except the verifier (View_S^Γ) as input, and outputs $sk' = sk$ with probability $p - \mu$, for negligible value of μ . This holds assuming, $\text{Com}(u;v)$ is a perfect binding computational hid-*

ing homomorphic bit commitment scheme (Definition 2.6.3), and ZKP is a κ -sound identification protocol (Definition 2.4.1).

Note that the extractor of this lemma, has a critical difference from the extractor that is considered in security analysis of ProProx scheme in the original paper [Vau14]; the input of the extractor of the original paper takes the view of the verifier is the view of all close-by participants, including the verifier, but the input of the above extractor is the view of all close-by participants (excluding the verifier). By excluding the view of the verifier of a TF attack from the view of the extractor, the close-by participants can extract the secret-key of the prover, even when the prover is using directional antenna to communicate directly to the verifier.

In the security analysis of the extractor of the original paper, it is assumed that a correct response is solely sent from a single close-by participant. However, there might be a case that the received message $r_{i,j}$ is the combination of a message that is sent from a far-away source and a message that is sent from a close-by source. In our extractor, we include this case.

Proof 12 (Extractor) *Let's assume there is a TF adversary \mathcal{A} that succeeds in Π protocol with non-negligible probability p , i.e., generates a transcript $\xi = (A, [c], [r])$ that is accepted by the verifier with probability p . We construct a PPT extractor algorithm \mathcal{E} for the secret key.*

In a protocol Π from game Γ , the sequence of all challenges $[c]$ (slow and fast) is chosen randomly and broadcasted by the honest verifier. We define $[r] = [r]^{fast} || [r]^{slow}$ and $[c] = [c]^{fast} || [c]^{slow}$ where the superscripts show the type of the phase of the challenges.

Because of the perfect binding commitment (Definition 2.6.2), the value of the public key pk uniquely determines $sk = Com^{-1}(pk)$, and the value of $A_{i,j}$ uniquely determines $a_{i,j} = Com^{-1}(A_{i,j})$. We emphasize that these values are not being calculate, but we just mathematically define them based on the view of the verifier.

Let S be the event that for all j , and $i \in I_j$, the verifier's checks $r_{i,j} = a_{i,j} + c_{i,j}b_i + c_{i,j}sk_j$ hold true. This can be verified by checking success of ZKP, for all the corresponding j and $i \in I_j$. In

other words, when ZKP succeeds for all j , and $i \in I_j$, we have $z_{i,j}$ as commitment to $a_{i,j} + c_{i,j}b_i + c_{i,j}sk_j - r_{i,j}$, which implies the occurrence of S .

Since ZKP is κ -sound, we conclude that $\Pr[\text{succ ZKP} | \neg S] \leq \kappa$ and then $\Pr[\text{succ ZKP}, \neg S] \leq \kappa$, where $\neg S$ indicates negation of S . So we have $\Pr[\text{valid } \xi, \neg S] \leq \kappa$.

Based on the fundamental lemma (Lemma 5), any valid response $r_{i,j}$ that is received by the verifier consists of two parts $r_{\mathcal{F}}$ and $r_{\mathcal{S}}$ that $r_{\mathcal{F}} = \text{Msg}_{\mathcal{F} \rightarrow v(t')}(c_{i,j}^-)$ and $\exists \mathcal{J}' : r_{\mathcal{S}} = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(c_{i,j}^-), c_{i,j}, \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(c_{i,j}^-))$. We have $r_{i,j} = \text{sum}(r_{\mathcal{F}}, r_{\mathcal{S}})$, for a deterministic function $\text{sum}()$ that is determined by the physical communication channel. We assume there exist a deterministic subtraction function $\text{sub}()$ such that for any $x = \text{sum}(z, y)$, we have $z = \text{sub}(x, y)$ and $y = \text{sub}(x, z)$.

We consider the view of close-by participants before sending the response $r_{i,j}$, i.e., $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{i,j})$, relative to the view of the close-by participants before seeing the challenge $c_{i,j}$, i.e., $\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{i,j})$. In the time period between receiving the challenge $c_{i,j}$ and sending $r_{i,j}$, the close-by participants can receive messages from two different sources: the verifier, and the far-away participants. The only message from the verifier in this period is $c_{i,j}$ and we indicate the messages from the far-away participants as $\text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{i,j})$ which is independent from $c_{i,j}$. So we have $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{i,j}) = \text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{i,j}) || c_{i,j} || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{i,j})$.

We conclude that there is an algorithm \mathcal{J}' that takes $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{i,j})$ and generates $r_{\mathcal{S}}$, such that the correct response $r_{i,j}$ is calculable as $r_{i,j} = \text{sum}(\text{Msg}_{\mathcal{F} \rightarrow v(t')}(c_{i,j}^-), r_{\mathcal{S}})$. Note that $\text{View}_{\mathcal{S}}^{\Gamma}(\neg r_{i,j})$ includes the challenge $c_{i,j}$.

Since there is an algorithm \mathcal{J}' that $\text{resp}_{i,j} = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{i,j}) || c_{i,j} || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{i,j}))$, then we construct the algorithm \mathcal{J} that calls \mathcal{J}' two times with the following inputs: $\text{resp}_{i,j}^0 = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{i,j}) || 0 || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{i,j}))$ and then $\text{resp}_{i,j}^1 = \mathcal{J}'(\text{View}_{\mathcal{S}}^{\Gamma}(\neg c_{i,j}) || 1 || \text{Msg}_{\mathcal{F} \rightarrow \mathcal{S}}(\neg c_{i,j}))$. As a result, \mathcal{J} returns a pair $(\text{resp}_{i,j}^0, \text{resp}_{i,j}^1)$ such that the output of the functions $\text{sum}(\text{resp}_{i,j}^0, r_{\mathcal{F}})$ and $\text{sum}(\text{resp}_{i,j}^1, r_{\mathcal{F}})$ are the correct answer for the two possible cases of the challenge bit $c_{i,j}$. We say $\text{resp}_{i,j}^0$ (or $\text{resp}_{i,j}^1$) is correct if $\text{sum}(\text{resp}_{i,j}^0, r_{\mathcal{F}})$ (or $\text{sum}(\text{resp}_{i,j}^1, r_{\mathcal{F}})$) is the correct response to the challenge $c_{i,j} = 0$ (or

$c_{i,j} = 1$). Note that the value of $r_{\mathcal{F}}$ is not known to the extractor.

From the final view of the close-by participants, we can find their partial view at the time of sending response $r_{i,j}$ for all $j = 1 \dots \lambda$ and $i = 1 \dots n$, and then call the algorithm \mathcal{J} . So we can calculate $(resp_{i,j}^0, resp_{i,j}^1)$ for all $j = 1 \dots \lambda$ and $i = 1 \dots n$, from the final view of the close-by participants. Since we have

$$sub(sum(resp_{i,j}^1, r_{\mathcal{F}}); sum(resp_{i,j}^0, r_{\mathcal{F}})) = sub(resp_{i,j}^1; resp_{i,j}^0),$$

therefore, calculating the difference between the two correct responses for both values of the challenge bit $c_{i,j}$ is equal to the difference between $resp_{i,j}^1$ and $resp_{i,j}^0$, which are computable from the final view of close-by participants ($View_s^\Gamma$).

We build the extractor \mathcal{E} as follows: \mathcal{E} runs $(resp_{i,j}^0, resp_{i,j}^1) \leftarrow \mathcal{J}(View_s^\Gamma)$ for all $j = 1 \dots \lambda$ and $i = 1 \dots n$. Then it guesses the bits of secret $sk'_j = \text{majority}(\delta_{1,j} \dots \delta_{n,j})$ for $j = \{1 \dots \lambda\}$, where $\delta_{i,j} = sub(resp_{i,j}^1; resp_{i,j}^0) - b_i$ for $i = \{1 \dots n\}$.

Note that in a simple case that the received message at receiver is sent from a single source, we have $sub(resp_{i,j}^1; resp_{i,j}^0) = resp_{i,j}^1 - resp_{i,j}^0$. In the following we calculate the success chance of this extractor;

A response $resp_{i,j}^d$ for $d \in \{0, 1\}$ is correct if $sum(resp_{i,j}^d, r_{\mathcal{F}}) = a_{i,j} + d.b_i + d.sk_j$. We define $R_j = [R_{1,j} \dots R_{n,j}]$ where $R_{i,j}$ is the number of challenge bits $d \in \{0, 1\}$ for which the response $resp_{i,j}^d$ is correct. If we have $R_{i,j} = 2$ then $\delta_{i,j} = sk_j$, but if $R_{i,j} = 1$ then we might have $\delta_{i,j} \neq sk_j$.

Consider $R = (R^1, \dots, R^\lambda)$ for vector $R^i = (R_{i1}, \dots, R_{in})$, where $R_{i,j}$ is defined as above and calculated by comparing the $(resp_{i,j}^0, resp_{i,j}^1)$ with correct responses for all $j = 1 \dots \lambda$ and $i = 1 \dots n$. We define the set \mathcal{R} as all vectors in $\{0, 1, 2\}^n$ that have at least $\lfloor \frac{n}{2} \rfloor + 1$ values of 2. For a vector $R_j \in \mathcal{R}$, we have a majority of i 's that $\delta_{i,j} = sk_j$, which implies $sk'_j = sk_j$.

Since the verifier selects the challenges randomly, then knowing $R_{i,j}$ allows us to find the probability that the response $r_{i,j} = sum(resp_{i,j}, r_{\mathcal{F}})$ is correct: if $R_{i,j} = 2$ then this probability is 1,

otherwise this probability is at most $\frac{1}{2}$ (this is true that a response can always be guessed randomly). If W is the random variable giving the number R_j s that $R_j \notin \mathcal{R}$, we have $\Pr[S|W = w] = p_B^w$ where $p_B = \text{Tail}(\lceil \frac{n}{2} \rceil, \tau - \lfloor \frac{n}{2} \rfloor, \frac{1}{2})$, defined in Theorem 1. So $\Pr[S, W = w] \leq p_B^w \Pr[W = w]$ and then $\Pr[S, W \geq w] \leq p_B^w$. As a result, we have:

$$\Pr[W \geq w, \text{valid}\xi] \leq \Pr[\neg S, \text{valid}\xi] + \Pr[S, W \geq w] \leq \kappa + p_B^w$$

Each index j where $sk_j \neq sk'_j$, corresponds to $R_j \notin \mathcal{R}$. Therefore, having the verifier outputting $\text{Out}_V = 1$ (i.e., ξ is valid) and the extractor giving at least 1 error occurs with probability bounded by $\mu = \kappa + p_B$. This implies that we can build a key extractor from View_S^Γ that follows the success chance of the TF attack, except with probability μ that is negligible because of Chernoff bound (Theorem 1). ■

Lemma 13 (TF-resistance) *ProProx is a μ -TF-resistant DBID (Property 4.2.4) scheme for negligible value of μ , assuming Com is a perfectly binding computational hiding homomorphic bit commitment (Definition 2.6.3), and ZKP is a κ -sound identification protocol (Definition 2.4.1).*

Proof 13 According to the TF-resistance definition (Property 4.2.4), we need to show that if there is a game Γ for a Π protocol in which that the verifier returns $\text{Out}_V = 1$ with non-negligible probability κ , then there exists a close-by actor \mathcal{R} that for any challenge sequence $[c]$ can create a valid transcript with probability at least $\kappa - \mu$ for a negligible μ , using the view of all close-by participants, excluding the verifier (View_S^Γ).

Based on Lemma 12, the existence of TF attacker with non-negligible success probability κ , implies the existence of a key extractor $sk' \leftarrow \mathcal{E}(\text{View}_S^\Gamma)$ with the success chance of at least $\kappa - \mu'$ for a negligible μ' .

So we make \mathcal{R} as follows: After a successful TF attack, \mathcal{R} calls the above extractor \mathcal{E} to find sk' . Then \mathcal{R} runs the $\text{ProProx}.\Pi.P(sk', gpk)$ interactive algorithm in order to generate valid transcript with correct timing for any challenge that is generated by the verifier. Since ProProx is (τ, δ) -

complete, the verifier outputs $\text{Out}_V = 1$, unless negligible probability δ . Therefore, the success chance of \mathcal{R} is at least $\kappa - \mu$ for negligible $\mu = \mu' - \delta$. \blacksquare

Lemma 14 (Soundness) *ProProx is a γ -sound DBID (Property 4.2.2) scheme for $\gamma = \text{negl}(\lambda)$, if the followings hold: $\tau.n \geq n - (\frac{1}{2} - 2\epsilon)\lceil \frac{n}{2} \rceil$ for some constant $\epsilon > 0$; Com_H is one-way; Com is homomorphic bit commitment with all properties of Definition 2.6.3; ZKP is a κ -sound (Definition 2.4.1) and ζ -zero-knowledge identification (Definition 2.5.4) for negligible κ and ζ .*

Proof 14 *In a Π protocol, the verifier receives a transcript $\xi = (A, [c], [r])$. There are two possible participant arrangements for the winning conditions of a soundness adversary that result in the verifier returning $\text{Out}_V = 1$: (i) all active provers are far-away from the verifier, (ii) there is no active prover during the Π protocol (i.e., there might be close-by provers but they are not active). In the following we show that the success probability of the adversary in both cases is negligible. In other words, the success probability of generating a valid transcript $\xi = (A, [c], [r])$ when the challenge sequence $[c]$ is generated by the verifier, is negligible.*

In the first case, the adversary cannot simply relay the messages because of the extra delay and the fact that the responses are from out of bound locations. In this case the verifier will reject the instance. If there is a PPT adversary \mathcal{A} that can guess at least $\tau.n$ out of n responses for each key bit with non-negligible probability (i.e., guessing all bits of $\forall j \in \{1, \dots, \lambda\} I_j \subset \{1, \dots, n\}$ such that $|I_j| \geq \tau.n$), then they can find the response table for at least $\tau.n$ elements for each $j \in \{1, \dots, \lambda\}$ with the same probability. So for $\tau.n$ out of n values of i they can find correct $sk_j = \frac{r_{i,j} - r_{i,j} - (c_{i,j} - \bar{c}_{i,j})b_i}{c_{i,j} - \bar{c}_{i,j}}$ with probability $\geq \text{poly}(\lambda)$. Therefore by taking the majority, they can find the correct key bit with probability $\geq 1 - (1 - \text{poly}(\lambda))^{\tau.n}$.

Thus if the adversary succeeds in the first case with non-negligible probability, then they can find the secret key with considerably higher probability than random guessing and this contradicts the zero-knowledge property of ProProx. Therefore, the adversary's success chance will be negligible in this case.

In the second case, the adversary succeeds in the protocol by providing the correct response to \mathcal{V} for at least $\tau.n$ correct queries out of n fast rounds for all key bits.

We noted that the learning phase of the adversary cannot provide information about the secret key $(\{sk_j\}_{j=1}^\lambda)$ or the committed values $(\{a_{i,j}\}_{j=\{1\dots\lambda\}, i=\{1\dots n\}})$ as otherwise the zero-knowledge property of the protocol, or the commitment scheme will be violated, respectively.

In order to succeed in the protocol with non-negligible probability, the adversary must succeed in ZKP, for at least $\tau.n$ values of i , so they need to find at least $\tau.n$ valid tuples $\pi_i = (X, Y, Z)$ for random challenge bits such that $Z^2 = X(\theta^{b_i}y_j)^{c_{i,j}}\theta^{-Y}$ without having information about sk_j . For $\pi = [\pi_i]$ with size at least $\tau.n$ and $[c]$, $\Pr[\pi \text{ is valid} | [c] \text{ is random}] = \prod_{i=1}^\tau \Pr[\pi_i \text{ is valid} | [c] \text{ is random}]$. So if $\Pr[\pi \text{ is valid} | [c] \text{ is random}] \geq \text{negl}$, then there is a value of i that $\Pr[\pi_i \text{ is valid} | [c] \text{ is random}] \geq \frac{1}{2} + \text{poly}(\lambda)$.

Since X is sent to the verifier before seeing $c_{i,j}$, therefore we have $\Pr[\text{valid}(X, Y, Z) | c_{i,j} = 0] \geq \frac{1}{2} + \text{poly}(\lambda)$ and also $\Pr[\text{valid}(X, Y', Z') | c_{i,j} = 1] \geq \frac{1}{2} + \text{poly}(\lambda)$. Since both tuples are valid, then we have $Z^2 = X\theta^{-Y}$ and $Z'^2 = X(\theta^{b_i}y_j)\theta^{-Y'}$. Therefore we have the following for $pk_j = \theta^{sk_j}(v_j)$:

$$\left(\frac{Z'}{Z}\right)^2 = pk_j \theta^{b_i - Y' + Y} = \theta^{sk_j + b_i - Y' + Y} (v_j)^2$$

Therefore, the adversary can conclude $sk_j + b_i - Y' + Y \notin \{1, 3\}$ for the known bits b_i , Y' and Y . So they gain some information about sk_j , which is in contradiction with zero-knowledge property of ProProx. ■

4.5 Related Works

The main models and constructions of public key DB protocols are presented in [HPO13], [ASN14], [GKL⁺14], and [Vau14]. In the following, we discuss and contrast the security model of these works to be able to put our new work in context.

[HPO13] presented an informal model for *Distance-Fraud*, *Mafia-Fraud* and *Impersonation* attack and provided a secure protocol according to the model. [ASN14] formally defined *Distance-Fraud*, *Mafia-Fraud*, *Impersonation*, *Terrorist-Fraud* and *Distance-Hijacking* attack. The *Distance-Fraud* adversary has a learning phase before the attack session and is therefore stronger than the definition in Attack 3.1.2. During the learning phase, the adversary has access to the communications of the honest provers that are close-by. The security proofs of the proposed protocol have been deferred to the full version, which is not available yet.

[GKL⁺14] uses an informal model that captures *Distance-Fraud*, *Mafia-Fraud*, *Impersonation*, *Terrorist-Fraud*, *Distance-Hijacking* and *Slow-Impersonation* [DFKO11]. In their model, the definition of *Terrorist-Fraud* is slightly different from Attack 3.1.3: a TF attack is successful if it allows the adversary to succeed in future Mafia-Fraud attacks.

For the first time in distance-bounding literature, [DFKO11] considered normal MiM attacking scenario where both the honest prover and the adversary are close to the verifier. The adversary interacts with the prover in order to succeed in a separate protocol session with the verifier. The adversary has to change some of the received messages in the slow phases of protocol in order to be considered successful. The attack is called *Slow-Impersonation* and is inspired by the basic MiM attack in authentication protocols. In *Slow-Impersonation*, a close-by MiM actor that communicates with both verifier and close-by prover, tries to succeed in the protocol. During the slow-phase, the actor modifies the received messages from a party, and then sends it to the other party. Although the basic MiM attack is proper for DB models, it may not be strictly possible in one phase of the protocol as their action could influence or be influenced by other phases of the protocol.

A MiM adversary may, during the learning phase, only relay the slow-phase messages but, by manipulating the messages of the fast phase, learn the key information and later succeed in impersonation. According to the definitions in [GKL⁺14] and [DFKO11], the protocol is secure against *Slow-Impersonation*, however it is not secure against *Strong-Impersonation* [ASN17b].

This scenario shows that *Slow-Impersonation* does not necessarily capture *Impersonation* attacks in general. Moreover, it's hard to distinguish the success in slow phases of a protocol without considering the fast phase, as those phases have mutual influences on each other.

As an alternative definition, [ASN17a] proposed *Strong-Impersonation*: a close-by actor learns from past executions of the protocol by a close-by honest prover and tries to impersonate the prover in a new execution when the prover is either inactive or is not close-by anymore.

In this attack, the MiM adversary has an active learning phase that allows them to change the messages. *Strong-Impersonation* captures the MiM attack without the need to define success in the slow rounds. One of the incentives of *Strong-Impersonation* is capturing the case when the prover is close to the verifier, but is not participating in any instance of the protocol. In this case, any acceptance by the verifier means that the adversary has succeeded in impersonating an inactive prover.

In [Vau14] an elegant formal model for public key distance-bounding protocols in terms of proof of proximity of knowledge has been proposed. The model captures *Distance-Fraud*, *Distance-Hijacking*, *Mafia-Fraud*, *Impersonation* and *Terrorist-Fraud*. In this approach, a public key DB protocol is a special type of proof of knowledge (proximity of knowledge): a protocol is considered sound if the acceptance of the verifier implies existence of an extractor algorithm that takes the view of all close-by participants and returns the prover's private key. This captures security against *Terrorist-Fraud* where a dishonest far-away prover must succeed without sharing their key with the close-by helper.

According to the soundness definition in [Vau14] however, if the adversary succeeds while there is an inactive close-by prover, the protocol is sound because the verifier accepts, and there is an extractor for the key simply because there is an inactive close-by prover and their secret key is part of the extractor's view. Existence of an extractor is a demanding requirement for the success of attacks against authentication: obviously an adversary who can extract the key will succeed in the

protocol, however it is possible to have an adversary who succeeds without extracting the key, but providing the required responses to the verifier. Our goal in introducing identification based model is to capture this weaker requirement of success in authentication, while providing a model that includes realistic attacks against DB protocols.

4.6 Concluding Remarks

We proposed a new formal model (DBID) for distance-bounding protocols, inline with the cryptographic identification schemes, that captures and strengthens the main attacks on public key distance-bounding protocols. This approach effectively included physical distance as an additional attribute of the prover in identification protocol. In our model for the first time, we allow the communication channel to mix multiple sent messages into one message at the receiver, if they are received at the same time. This is a more realistic wireless environment.

In this model, we assume a stronger adversary that has access to a directional antenna. We showed this additional capability can break security of protocols that had been proven secure, by proposing directional TF attack against VSSDB and DBPoK- \log^+ schemes. To include this capability of the adversary, we needed to revise the definition of TF in [ASN17a] which resulted in proving a new security proof for the ProProx protocol.

We showed that the ProProx scheme fits in our model and proved its security. We also proposed Poxxy, as a new DBID scheme, and provided the security proof. Poxxy is an advanced edition of DBPoK- \log^+ , that we proposed earlier in this research [ASN14]. Poxxy and ProProx use two different cryptosystems, and the fast phase of the Π protocol in Poxxy and ProProx schemes have more computations compared to typical DB schemes. This is the price we pay for achieving TF-resistance.

Some parts of this chapter are published in [ASN14, ASN17a, ASN17b].

Chapter 5

Anonymous Distance-Bounding

Anonymous Distance-Bounding (DB) protocols allow a prover to convince a verifier that they are within a distance bound from them, without revealing their identity. This is an attractive property that enables the prover to enjoy proximity based services, while their privacy is maintained. Combination of anonymity and distance-bounding however introduces new security challenges. In this chapter, we consider two new realistic attacks: a physical layer attack that uses *directional antenna*, and a *collusion attack* that involves multiple users. We show all existing anonymous DB protocols become insecure against at least one of these attacks, and then propose a new security model that captures these new attacks, and finally construct two protocols with provable security in this model. Our protocols are the only known anonymous DB protocols with provable security against known attacks.

Distance-bounding protocols leak the identity of prover to verifier: in symmetric key DB, prover and verifier share a secret key, and in public-key DB, prover's response is compared against the public-key of a specific user. Anonymous DB can be used to prove that the distance of a registered user is less than a prescribed bound, without revealing their exact identity. Security of anonymous DB protocols has also been formalized [ASN14, BGG⁺16, ABG⁺17] against DF, MF and TF. In these model that we refer to as *single-user* model, attack involves at most a single corrupted user, possibly helped by an actor.

Our work: We introduce two new type of attacks that although applicable to all DB protocols, become particularly effective against anonymous DB protocols.

Directional antennas. The use of directional antennas in consumer devices has grown tremendously in recent years [ARS16]. We consider the effect of employing these antennas by a malicious prover, on the security of anonymous DB. Note that verifiers need to use omni-directional antennas because they want estimate the distance of the prover without knowing their location. However malicious provers (or actors) may use directional antennas with a narrow beam to target messages to the verifier. In Section 5.1.1 we show that using directional antennas by malicious provers can break all existing anonymous DB protocols.

Collusion attacks. In a collusion attack multiple users, each with a secret key, participate in the attack that can be on DF, MF or TF form as shown in Figure 5.1, Figure 5.2 and Figure 5.3. These attacks had not been considered before and need not to be considered as long as the secret keys of two users are independently generated, and so (without anonymity) a protocol transcript can be linked to a user through their key information and so cannot be combined with other transcripts to form a new forged transcript. In anonymous DB protocols however, the verifier should not be able to link the transcript of a protocol to a single user and so combining protocol transcripts can give advantage to colluders. In Section 5.1.2 we show that collusion TF attack can be used to subvert traceability property of anonymous DB. This functionality is necessary in all anonymous DB protocols to ensure user accountability by allowing a third party that holds a master key, to "open" a transcript and identify the user, when required.

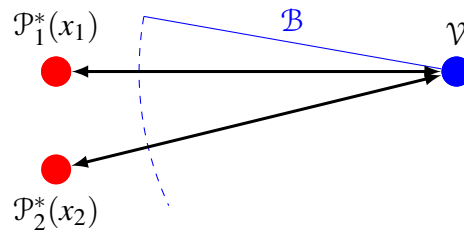


Figure 5.1: Collusion DF

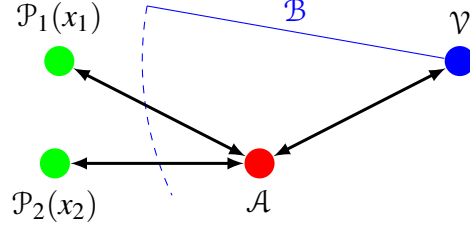


Figure 5.2: Collusion MF

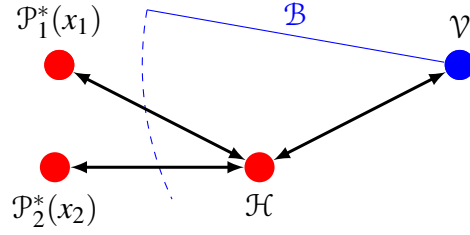


Figure 5.3: Collusion TF

The above two classes of attacks are realistic. Directional antennas are widely used in modern communication systems [ARS16] and there are strong incentive to launch collusion attacks, hiding behind anonymity that is offered by anonymous DB protocols. None of the existing anonymous DB protocols provide security against these attacks, and as shown later, there are concrete attacks against all existing protocols.

We will then show how collusion TF attack succeeds on a modified version of SPADE that is secure against a single-user non-directional TF attack. In this attack, a close-by user can interact with the verifier and get accepted, while credentials of a far-away user is used. Thus the close-by user can be authenticated, and later during the opening phase a far-away user be identified. The system fails to provide security because the far-away user can present alibi that they have not been the protocol participant.

Model. We propose a formal model that captures the above two new classes of attacks. Our formalization uses a cryptographic approach and models an anonymous DB protocol as a cryptographic identification protocol [DDP06] where the prover, in addition to proving their cryptographic cre-

dentials, prove that they are within a distance bound from the verifier. This builds on the model of public-key DB [ASN17a], by including directional antennas and collusion DF, MF and TF.

We formalize anonymity in terms of indistinguishability of candidate provers, given the protocol transcript. The challenge is to include sufficient information about the user in the transcript to allow a third party that holds the master key be able to open the transcript and identify the user.

Construction. We construct two anonymous DB protocol and prove their security in our proposed model. Our constructions can be seen as modular construction that adds anonymity and security in the new model by introducing an additional layer of group identification to a public-key DB protocol with provable security in a single-user model. The proposed protocols consist of a phase in which the prover commits to a temporary public-key, followed by a public-key DB. These protocols are designed for two different cryptosystems; Goldwasser-Micali cryptosystem [GM84] and Pedersen commitment [Ped92].

We refer to DBID schemes (Definition 4.2.1) with the property as these classes of DBID schemes as DBID^{GM} and DBID^{P} , respectively. An example of DBID^{GM} scheme is ProProx [Vau14] and an example of DBID^{P} scheme is POXY (Section 4.3).

Organization. Section 5.1.1 proposes new directional TF attack that breaks all anonymous DB protocols and Section 5.1.2 proposes collusion DB attacks, generalizing traditional DB attacks. Section 5.2 presents our model, Section 5.3 Section 5.4 give the constructions and security proofs. Section 5.5 gives a summary of related works, and Section 5.6 concludes the chapter.

5.1 New Attacks

We present two new classes of attacks and show their effectiveness on anonymous DB protocols.

5.1.1 Directional TF Attack on Anonymous DB

DB protocols consist of - *slow phases* that are used during protocol initialization, and the final verification, and a *fast challenge-response phase* that is used for time (and so distance) measurement. Using a directional antenna a malicious prover can target the messages of the two phases such that the initialization messages are only received by the verifier and not the helper. This strategy allows the prover to send the whole challenge and response table of a particular protocol run to the helper, and so take advantage of the location of the helper, without leaking their long term key and so succeed in TF attack. Note that the prover is not leaking its identity to the helper. Figure 4.1 shows a directional TF, where the helper \mathcal{H} does not receive slow phase messages sent by a malicious prover P^* to V using a directional antenna (orange ribbon in Figure 4.1). Before the start of the fast-phase, P^* sends the fast challenge-response table to \mathcal{H} , making \mathcal{H} in-charge of responding to the fast-phase challenges.

In the following we describe how this setting helps a malicious prover to succeed in terrorist-fraud against all known anonymous DB protocols: PDB [ASN14], SPADE [BGG⁺16] and TREAD [ABG⁺17].

Directional TF on PDB [ASN14]

The presented model of [ASN14] follows the original definition of TF (Attack 3.1.3), and so our attack can be seen as outside their model. However, we showed that the original definition of TF is not suitable for anonymous distance-bounding protocols. In this section we present a TF attack against PDB, using the more recent definition of TF (Attack 3.1.3).

Π protocol of PDB: The Π protocol in PDB scheme consists of the following five steps between the verifier (V) and the prover (P). At the end of each step, the state of each of the two parties become the starting state of the next step.

Step (i) SPK is a signature proof-of-knowledge protocol, in which the prover uses the secret key

sk_i and a secret membership certificate (σ_i), and the verifier uses group public-key (gpk), as inputs. At the end of the protocol, the verifier will be convinced that the prover has a valid secret-key sk_i and membership certificate σ_i , in zero-knowledge (*i.e.*, the verifier doesn't learn anything about the prover's secrets). The verifier also learns the value of a commitment $C = g_1^{sk_i} \cdot g_2^r$, that will be used in the last step. This protocol uses the BBS⁺ signature scheme [CL04] for generating the membership certificate σ_i and the SPK protocol.

Step (ii) Bit Commitment (Figure 4.2) is a commitment protocol, in which the prover uses the secret key sk_i , and the verifier uses the group public-key (gpk), as input. In this protocol, the prover decides on the "fast challenge-response table" and commits to each bit in the table. The verifier learns the committed values of every single bit of the fast challenge-response table. This table consists of two rows: $\{r_b[l]\}_{l=\{1,\dots,\lambda\}, b \in \{0,1\}}$, where $r_b[l]$ is the response in the i^{th} fast challenge-response round. The corresponding committed values are $\{C_b[l]\}_{l=\{1,\dots,\lambda\}, b \in \{0,1\}}$, and the corresponding randomness of commitments are indicated by $\{v_b[l]\}_{l=\{1,\dots,\lambda\}, b \in \{0,1\}}$, where $v_b[l] \in \mathbb{Z}_p^*$. The commitment function is as follows: $C_b[l] = g_1^{r_b[l]} \cdot h^{v_b[l]}$ for $b \in \{0,1\}$, $l = \{1 \dots \lambda\}$, and $g_1, h \in \mathbb{Z}_p$. The committed table and the randomness is kept secret at the prover, while the commitments are sent to the verifier.

Step (iii) Fast Challenge/Response (Figure 4.3) is the distance-bounding protocol, in which the prover uses the calculated "fast challenge-response table" $\{r_b[l] : l = \{1 \dots \lambda\}, b = \{0,1\}\}$, generated in step (ii), as input.

Step (iv) Commitment Opening (Figure 4.4) is used to open half of the commitments, that correspond to the challenge bits sent by the verifier in step (iii). In this step, the prover uses the secret commitment randomness (*i.e.*, $\{v_b[l] : l = \{1 \dots \lambda\}, b = \{0,1\}\}$) and the challenge values of step (iii) (*i.e.*, c'), and the verifier uses the committed values (*i.e.*, $\{C_b[l] : l = \{1 \dots \lambda\}, b = \{0,1\}\}$) and the challenge and response values of step (iii) (*i.e.*,

c and r') as input. This protocol improves the original PDB protocol [ASN14] by becoming noise resistance. This step succeeds, if the noise counter is less than the threshold (i.e., $count_{noise} < \tau$).

Step (v) Proof-of-Knowledge is a protocol for zero-knowledge proof of equality that shows the secret key of step (i) and the committed secret key of step (ii) are the same. In this protocol, the prover uses the secret key sk_i and the commitment randomness of step (i) and step (ii), and the verifier uses the committed values of step (i) and step (ii), as input. C is the committed value of step (i), r is the commitment randomness of step (i), z is the accumulation of the committed values of step (ii) as $z = \prod_{l=1}^{\lambda} (C_0[l]C_1[l])^{2^{l-1}} \bmod p$, and v is the accumulation of the commitment randomness of step (ii) as $v = \sum_{l=1}^{\lambda} 2^l \cdot (v_0[l] + v_1[l]) \bmod (p-1)$.

For security parameter t , this protocol runs t iterations of zero-knowledge proof-of-knowledge (Definition 2.5.4 and Definition 2.5.3) where z and C satisfy the following relation: $PoK[(sk_i, v, r) : z = g^{u \cdot sk_i} \cdot h^v \wedge pk_i = g^{sk_i} \cdot g_2^r]$.

If all steps terminate successfully, then the verifier outputs $Out_V = 1$.

Lemma 15 *In the Π protocol of PDB scheme, the fast challenge-response table does not leak information about the membership certificate σ_i of the prover, unless negligible probability.*

Proof 15 *We know that by having the fast challenge-response table, we can calculate the secret-key of the prover, i.e., $sk = \frac{k+e}{u} \bmod (p-1)$ where k is fresh randomness. Note that the fast challenge-response table is the output of random function that takes sk as input. So it cannot leak any information about other independent secrets of the prover.*

A valid membership certificate σ is the signature of registration authority on the secret-key sk . If there is an adversary \mathcal{A} that can calculate the membership certificate from the secret-key sk , then \mathcal{A} is a successful forgery adversary of the signature scheme. Therefore, since we assume the signature scheme is forgery resistant, then the success chance of \mathcal{A} is negligible. ■

Attack 5.1.1 *P sends to V the slow phase messages of step (i) and step (ii), using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table $((r_0[i], r_1[i]))$ for $i = 1 \dots \lambda$ to \mathcal{H} , allowing \mathcal{H} to respond to V's challenges. P sends to V the slow phase messages of step (iv) and step (v), using directional antenna. In this way, the helper can respond in time and correctly to the challenges of the verifier during the fast challenge-response rounds. This attack makes the verifier to accept the protocol. Note that the fast challenge-response table does not leak the membership certificate σ_i , according to Lemma 15.*

Since the helper has no information about σ_i , it cannot succeed in step (i) of future impersonation attacks. Therefore, it cannot impersonate the prove in future, which is required for a successful TF attack (See Property 5.2.4).

Directional TF on SPADE

SPADE [BGG⁺16] is an anonymous DB system that use a group signature $G\text{Sign}_{sk_p}()$ to register users in an authorized group. A registered user can use their credentials to participate in the protocol without leaking their identity, hence ensuring anonymity. Figure 3.14 presents the Π distance bounding protocol of SPADE scheme.

Lemma 16 *In the Π protocol of SPADE scheme, the fast challenge-response table does not leak information about the secret of prover.*

Proof 16 *The fast challenge-response table is $r_i = \begin{cases} a_i & \text{for } i = \{1, \dots, \lambda\}. \\ a_i \oplus N_{\mathcal{P}i} \oplus m_i & \end{cases}$. In each instance of Π protocol, $N_{\mathcal{P}}$ and m are fresh and chosen randomly, and $a = \text{PRF}(N_{\mathcal{P}}, N_{\mathcal{V}})$ is the output of a pseudo-random function, fed with two fresh random inputs. Therefore, this table is independent from the secret value $sk_{\mathcal{P}}$. ■*

Attack 5.1.2 *P sends to V the slow phase message e to V using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table $((a_i, a_i \oplus N_{\mathcal{P}i} \oplus m_i))$ for $i = 1 \dots \lambda$*

to \mathcal{H} , allowing \mathcal{H} to respond to V 's challenges. The collusion of P and \mathcal{H} makes V to accept (i.e., $Out_V = 1$) and this is without P sending to \mathcal{H} any information that is dependent on the secret key of P (i.e., sk_P). Note that the secret key of P is required for generation of the message e which will not be known by \mathcal{H} .

According to Lemma 16, the fast challenge-response table does not leak any information about the prover's long-term secret sk_P . Since the helper has no information about sk_P , it cannot generate a valid message e in future, as the secret of prover is required to generate it. So its' success chance in a future impersonation attack will not be improved. This completes a successful TF attack (See Property 5.2.4).

Directional TF on TREAD

TREAD [ABG⁺17] is an anonymous DB system that use a group signature $GSign_{sk}()$ to register users in an authorized group. The verifier needs to be registered and have a key-pair of their own. The structure of TREAD is very similar to SPADE. A registered user can use their credentials to participate in the protocol without leaking their identity, hence ensuring anonymity. Figure 3.15 presents the Π distance bounding protocol of TREAD scheme.

Lemma 17 *In the Π protocol of TREAD scheme, the fast challenge-response table leaks no information about the secret of prover.*

Proof 17 The fast challenge-response table is $r_i = \begin{cases} \alpha_i & \text{if } c_i = 0 \\ \beta_i \oplus m_i & \text{if } c_i = 1 \end{cases}$ for $i = \{1, \dots, \lambda\}$. In each instance of Π protocol, β , α and m are fresh and chosen randomly. Therefore, this table is independent from the secret value sk_P . ■

Attack 5.1.3 P sends to V the slow phase message $e || if_{pub}$ to V using directional antenna. Before the start of the fast phase, P sends the fast challenge-response table $((\alpha_i, \beta_i \oplus m_i)$ for $i = 1 \dots \lambda$) to \mathcal{H} , allowing \mathcal{H} to respond to V 's challenges. The collusion of P and \mathcal{H} makes V to accept (i.e.,

$Out_V = 1$) and this is without P sending to \mathcal{H} any information that is dependent on the secret key of P (i.e., sk). Note that the secret key of P is required for generation of the message e which will not be known by \mathcal{H} .

According to Lemma 17, the fast challenge-response table does not leak any information about the prover's long-term secret sk . Since the helper has no information about sk , its' success chance in a future impersonation attack will not be improved, as the secret of prover is required to generate a valid message e . So its' success chance in a future impersonation attack will not be improved. This completes a successful TF attack (See Property 5.2.4).

Concluding Remarks on Directional TF

In all existing anonymous DB protocols, the fast challenge-response table does not determine the prover's credential with overwhelming probability. Directional TF attack allows the prover to limit the view of helper to the fast challenge-response table and so TF succeeds because the leaked information to the helper, does not allow the helper to succeed in a future attack individually, as required by the definition of TF attacks (see Property 5.2.4).

5.1.2 Collusion TF on Anonymous DB

In symmetric and public-key DB protocol attacks, collusion of a single registered user and an actor (non-registered user) is considered. In these DB protocols, there is no need to consider collusion of multiple registered users, as their keys are generated independently and each protocol session is linkable to a single user. However in anonymous DB, different user keys are dependent and protocol sessions are not linkable to a user by verifier. Here we show that in anonymous DB protocols collusion of multiple registered users must be considered also.

We consider two types of *collusion TF* attacks:

Attack 5.1.4 *In collusion TF type 1 attack, both colluding users are outside the bound and use a*

helper that is inside the bound (See Figure 5.4).

Attack 5.1.5 In collusion TF type 2, the helper can be a prover of a user, that tries to help the far-away provers of another user (See Figure 5.5).

Note that in type 2 attack there is a close-by prover \mathcal{P}_2^* who can succeed in the protocol by themselves. However by colluding with \mathcal{P}_1^* , can succeed without being traced! (This attack also works in public-key DB protocols such as [BB04], where users choose their own private-keys, and so can collude and choose related keys that leads to the success of the above attack.)

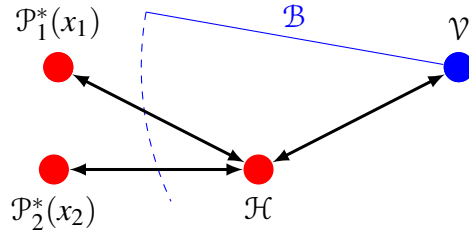


Figure 5.4: Collusion TF Attack Type 1

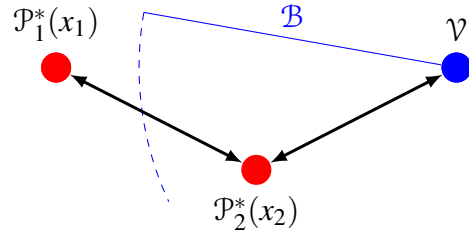


Figure 5.5: Collusion TF Attack Type 2

Both collusions can be used to increase the success chance of the attacker. Here we show how Collusion TF Type 2 (Figure 5.5) can break a protocol that is secure against TF in a single-user security model. As noted in Section 5.1.1, all existing anonymous DB protocols are vulnerable to single-user TF attack (directional TF) and so to show that protection against single-user TF attack does not imply security against collusion TF attack, we first modify SPADE protocol to make it (intuitively) secure against single-user TF attacks (given in Section 5.1.1), and then describe how a multi-user collusion TF attack succeeds against the modified protocol.

SPADE* (*modified* SPADE). We modify the challenge-response table of the SPADE protocol to the following: $r_i = \begin{cases} a_i & \text{if } c_i = 0 \\ a_i \oplus x_i & \text{if } c_i = 1 \end{cases}$, where x is part of the prover secret-key that is chosen independent of $sk_{\mathcal{P}}$, and $|x| = \lambda$. The verification phase will also be revised to accommodate this change and allow the verifier can check if the correct parameters are used in the challenge-response table. The challenge-response table of SPADE* contains the secret-key of the prover, which makes the protocol intuitively secure against single-user TF attacks (let's assume that). This if the whole table is leaked to the helper, the helper can learn the secret key of the (malicious) prover by XOR-ing the two response bits of each challenge. Now we propose a collusion TF Type 2 (Figure 5.5) against SPADE*;

Attack 5.1.6 (Collusion TF Type 2 Attack against SPADE*) *First, $\mathcal{P}_1^*(x_1)$ runs the "Initialization" phase of SPADE* with the verifier from outside the distance bound, and sends a to $\mathcal{P}_2^*(x_2)$. Then $\mathcal{P}_2^*(x_2)$ runs the challenge-response and verification phase with the verifier from inside the distance bound with its own credentials (x_2).*

The intuition for the attack is that the challenge-response table is not linked to the long-term secret key of the user (group signature key). The verifier sees σ which is the group signature of the far-away prover \mathcal{P}_1^ , but runs the distance bounding phase using a key that is not related to group signature key. Thus the tracing authority will link the session to x_1 , which is a violation of TF-resistance (Property 5.2.4).*

5.2 Model

Firstly we define the settings of the system, *i.e.*, entities and how they communicate, protocol and view of an entity, adversary and their capability. Then we provide a definition of anonymous distance-bounding (AnonDB) and also describe AnonDB experiment, which captures an AnonDB scheme in execution. Finally, we formalize six security properties (*Completeness, MiM-resistance,*

DF-resistance, Soundness, Traceability, Anonymity) of anonymous distance-bounding systems based on a game (AnonDB game), which is an AnonDB experiment played between a challenger and an adversary.

Relation with Public-Key DB Model: This model (Called AnonDB) is an extension of our model about public-key DB (Section 4.2). The main difference between these models is the fact that verifier does not know the identity of prover. This fact allows multiple provers to collude in attacks against the verifier. Moreover, because of anonymity of provers, a new definition of TF attack is considered.

Entities. There are m users in the system $\mathcal{U} = \{u_1, \dots, u_m\}$. Each user in the system can have multiple provers, which captures the practical scenario of a single person having multiple devices. We denote the list of provers for a user u_i as \mathcal{P}^i . Thus, we have m list of provers forming the prover set $\mathcal{P} = \{\mathcal{P}^1, \dots, \mathcal{P}^m\}$.

A trusted group manager generates the public parameters of the system, registers users and issues a unique group membership certificate to each user. A user u_i ($1 \leq i \leq m$) is identifiable by their certificate. The certificate, that must be kept secret, forms the secret input of the user in proving their membership in the group. The certificate of user u_i is shared by all provers of the list \mathcal{P}^i .

We define three types of *participants* in the system: provers (\mathcal{P}), verifiers (\mathcal{V}) (a singleton set), and actors (\mathcal{T}), called *helpers in TF attack*. \mathcal{V} and \mathcal{T} have access to only the public parameters of the system. Each participant has a location $loc = (x, y) \in \mathbb{R} \times \mathbb{R}$, that is an element of a metric space equipped with Euclidean distance, and is fixed during the protocol. The distance function $d(loc_1, loc_2)$ returns the distance between any two locations. Message travel time between locations loc_1 and loc_2 is $\frac{d(loc_1, loc_2)}{c}$, where c is the speed of light. A bit sent over the channel may flip with probability p_{noise} ($0 \leq p_{noise} \leq 1$). Participants, if located within a predefined distance bound \mathcal{B} from \mathcal{V} , are called *close-by* participants, otherwise they are called *far-away* participants.

Communication Structure. Each participant is equipped with a directional and an omni-directional

antennas. Having directional antennas enables them to choose the angle of the transmission beam such that only the intended participants receive the message.

View. The view of an entity at any point (in time) of a protocol consists of: all the inputs of the entity (including random coin tosses) and the set of messages received by that entity up to that point. Any instance of receiving message is called an *event*. $View_x^\Gamma(e)$ is a random variable that denotes the view of an entity x right after the event e in protocol Γ . $View_x^\Gamma$ denotes the view of x at the end of the protocol Γ , i.e., $View_x^\Gamma = View_x^\Gamma(e_{last})$ where e_{last} is the last event in Γ .

Adversary. An adversary can corrupt any subset of participants $\mathcal{X}^* \subset \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$. Corrupting one prover from a prover subset (e.g., $x \in \mathcal{P}^j$) effectively corrupts the whole subset, since all members of that subset share the same certificate (of user u_j). Provers of uncorrupted subset follow the protocol, and only one prover from the subset executes the protocol at a time. We do not restrict provers of corrupted subset from doing this.

For each security property, the adversary has certain goals, which is reflected as restrictions of \mathcal{X}^* ;

- in *Completeness* $\mathcal{X}^* = \emptyset$,
- in *Soundness* $\mathcal{X}^* \subseteq \mathcal{T}$,
- in *DF-resistance* $\mathcal{X}^* \subseteq \mathcal{P}$,
- in *TF-resistance* and *Traceability* $\mathcal{X}^* \subseteq \mathcal{P} \cup \mathcal{T}$, and
- in *Anonymity* $\mathcal{X}^* \subseteq \mathcal{V} \cup \mathcal{T}$.

Below we use the approach of [ASN17a] to define AnonDB scheme.

Definition 5.2.1 (Anonymous Distance-Bounding Scheme) For a security parameter λ , an anonymous distance-bounding (AnonDB) scheme is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathbb{B}, p_{noise}, \text{Init}, \text{CertGen}, \text{CertVer}, \Pi, \text{Revoke}, \text{Open})$, where;

- (I) \mathbb{X} and \mathbb{Y} are sets of possible system master keys and group public-keys, respectively.

$\text{Init}(1^\lambda)$ is the function that the group manager uses to generate the system master key msk , and the group public-key gpk .

(II) \mathbb{S} is set of possible user membership certificates.

$\text{CertGen}(1^\lambda, \text{msk}, \text{gpk}, i)$ generates a user membership certificate s_i , and $\text{CertVer}(s_i, \text{gpk})$ validates a user's certificate with respect to the group public-key.

(III) Π is a DB protocol between prover $P(s_i, \text{gpk})$ and verifier $V(\text{gpk})$, in which V verifies that a group member is located within the distance bound \mathcal{B} to the verifier.

(IV) The transmitted bits of a fast challenge-response round is affected by noise where $p_{\text{noise}} \in [0, 1]$ is the probability of a bit flip on each fast challenge-response message.

(V) $\text{Revoke}(\text{msk}, \text{gpk}, i)$ is an algorithm that removes a user (u_i) from the system and updates the master secret-key and group public key accordingly.

(VI) $\text{Open}(\text{msk}, \text{View}_V^\Pi)$ is an algorithm that identifies the user (u_i) that is involved in the Π protocol, using view of the verifier.

The operations Open and Revoke are optional in an AnonDB scheme. Note that $I - V$ above is the same as $I - V$ in DBID scheme (Definition 4.2.1), with the only difference that in DBID, each user owns a key-pair, while in AnonDB, each user owns a membership certificate that allows them to prove their membership according to the group public-key.

Adversary's capability is modeled as their access to queries presented to the challenger. The security properties of an anonymous DB protocol are based on a game (AnonDB Game) between a challenger and an adversary. Note that we allow provers to have access to directional antenna (to captures the directional attack introduced in Section 5.1.1), and presence of multiple, possibly colluding users (with different secret keys) in the system (to capture multiple user collusion attack introduced in Section 5.1.2).

Below we describe a general execution of an instance of the AnonDB scheme, which we call AnonDB

experiment. And after that, we define AnonDB game as an special AnonDB experiment.

Definition 5.2.2 (AnonDB Experiment) *An AnonDB experiment is defined by a tuple $(\text{AnonDB}; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T})$, where*

- (i) *AnonDB is an anonymous distance-bounding scheme as defined in Definition 5.2.1.*
- (ii) *\mathcal{U} is the set of users that are members of the group; each user $u_j \in \mathcal{U}$ has the following attributes:*
 - *$u_j.\text{Cert}$ that is a secret group membership certificate generated by the group manager,*
 - *$u_j.\text{RT}$ that is the registration time of the user that can be any time, and*
 - *$u_j.\text{Rev}$ that is a flag that shows if the user is revoked.*
- (iii) *\mathcal{P} is the set of provers; each prover has access to the membership certificate of a single user.*
- (iv) *\mathcal{V} is the set of verifiers; that have access to the group public-key of the AnonDB system. We consider the case where \mathcal{V} has a single member.*
- (v) *\mathcal{T} is the set of actors; each actor has access to the group public key of the AnonDB system.*

Members of the set $\mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ are called participants of the system. Each of the participants $x \in \mathcal{X}$ has the following attributes:

- a1. $x.\text{Loc}$ is the location of the participant,*
- a2. $x.\text{Code}$ is the code to run by the participant,*
- a3. $x.\text{St}$ that is the start time of the $x.\text{Code}$ execution, and*
- a4. $x.\text{Corr}$ is a flag indicating if the participant is corrupted or not.*

In addition to these attributes, each prover $p \in \mathcal{P}$ has one extra attribute:

- a5. $p.\text{Key}$ that is the secret certificate of the corresponding user, i.e., $p.\text{Key} = u_j.\text{Cert}$ for user $u_j \in \mathcal{U}$.*

All the provers that share the same certificate are called a prover subset, i.e., $\mathcal{P}^j = \{p : p \in \mathcal{P}, p.Key = u_j.Cert\}$. The start time of all provers is after registration time of the corresponding user, i.e., $\forall u_j \in \mathcal{U}, \forall p \in \mathcal{P}^j : p.St > u_j.RT$.

Members of a prover subset are either all honest or all dishonest. i.e., $\forall \mathcal{P}^j \in \mathcal{P}, \forall p \in \mathcal{P}^j : p.Corr = flag$, where $flag \in \{true, false\}$. All members of an honest prover subset $p \in \mathcal{P}^j$ follow the Π protocol (i.e., $p.Code = \text{AnonDB}.\Pi.P(.)$) and there is no overlap in the execution time of the members of an honest prover subset. If the verifier is honest, then it follows the Π protocol (i.e., $v.Code = \text{AnonDB}.\Pi.V(.)$ for $v \in \mathcal{V}$).

The experiment is run by a simulator that sets the attributes of the participants, and interacts with the group manager to assign keys to the provers of a user. If there is an adversary in the system, it interacts with the simulator to influence the experiment.

The experiment, without any adversary, proceeds as follows:

1. Setup:

(a) **Initialize:** The group manager runs $(msk/gpk) \leftarrow \text{AnonDB.Init}(1^\lambda)$ algorithm to generate the master secret-key and group public-key.

(b) **Generate Players:** The simulator forms the sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ and sets their attributes. The simulator interacts with the group manager obtain and assign keys of the provers. It also simulates the behavior of malicious players by setting their code ($x.Code$).

2. **Run:** The simulator starts the execution of $x.Code$ for all participants $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ at time $x.St$.

We assume the existence of a system clock that assigns time to events. The start and finish time of a protocol Γ is denoted as $stTime(\Gamma)$ and $fshTime(\Gamma)$ respectively, which form the execution time $exTime(\Gamma) = (stTime(\Gamma), fshTime(\Gamma))$ as a range of time and the execution length $exLen(\Gamma) = fshTime(\Gamma) - stTime(\Gamma)$. Members of a prover list $(\mathcal{P}^i, 1 \leq i \leq m)$ have different execution time

period (*i.e.*, they participate in a protocol from time t_1 to t_2), and possibly different locations.

In the following, we define a game between challenger and an adversary. This game is a limited AnonDB experiment that is run by the challenger who interacts with an adversary. In this game, the challenger plays both roles of the simulator and the group manager in the AnonDB experiment (Definition 5.2.2). The adversary's capabilities is modeled as access to a query that it presents to the challenger.

Definition 5.2.3 (AnonDB Game) *An AnonDB game between a challenger and adversary is an AnonDB experiment that is defined by a tuple $(\text{AnonDB}; \mathcal{U}; \mathcal{P}; \mathcal{V}; \mathcal{T}; \text{CorruptParties})$ where*

- *AnonDB is an anonymous distance-bounding scheme as defined in Definition 5.2.1.*
- *$\mathcal{U}, \mathcal{P}, \mathcal{V}$ and \mathcal{T} are the sets of users, provers, verifiers and actors, as defined in Definition 5.2.2, that are determined through interaction of the challenger and the adversary.*
- *$\text{CorruptParties}(Q)$ is a query that allows the adversary to plan their attack. Q is a set of participants, that may exist in the system or be introduced by the adversary.*

In more details:

1. Setup:

(a) **Initialize:** *Challenger runs $(\text{msk}/\text{gpk}) \leftarrow \text{AnonDB.Init}(1^\lambda)$ and publishes gpk . Note that the execution codes of honest prover and verifier are known by the challenger and the adversary at this point, and referred as $\text{AnonDB}.\Pi.P$ and $\text{AnonDB}.\Pi.V$ respectively.*

(b) **Generate Players:** *The sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ are formed through the interaction of the challenger and the adversary:*

i. The challenger creates the sets $(\mathcal{U}, \mathcal{V}, \mathcal{P}, \mathcal{T})$ as follows:

- *$\mathcal{V} = \{v\}$, where:*

a1. $v.Loc = loc_0$,

a2. $v.Code = \text{AnonDB}.\Pi.V$,

a3. $v.St = 0$, and

a4. $v.Corr = \text{false}$.

- $\mathcal{U} = \{u_j\}_{j=\{1,\dots,m\}}$, where $u_j.Cert$ is generated by $\text{AnonDB.CertGen}(1^\lambda, msk, gpk, j)$ function. The registration time of the users are set as $u_j.RT = 0$ and their revocation flag is set as $u_j.Rev = \text{false}$.

- $\mathcal{P} = \cup_{j=1}^m \mathcal{P}^j$, where \mathcal{P}^j is created as the prover subset of $u_j \in \mathcal{U}$. For all $p \in \mathcal{P}_{\{j=1\dots m\}}^j$ assigns their attributes:

a1. $p.Loc$ is set arbitrarily,

a2. $p.Code = \text{AnonDB}.\Pi.P$,

a3. $p.St$ is set arbitrarily such that there is no overlap in the execution time of the provers in \mathcal{P}^j ,

a4. $p.Corr = \text{false}$, and

a5. secret-key $p.Key = u_j.Cert$.

- $\mathcal{T} = \emptyset$

ii. The challenger sends the attributes $(x.Loc, x.Code, x.St)$ for all $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$, along with all prover subsets $\mathcal{P}^j \in \mathcal{P}$ to the adversary. The size of the set \mathcal{X} is n .

iii. The adversary generates $\text{CorruptParties}(Q)$ query and sends to the challenger. The challenger sends the secret information of the corrupted participants in Q to the adversary and their behavior (Code, Location and Start Time) is assigned according to the adversary instruction and their corruption flag is set to True. For all values of $j = 1\dots m$, if any prover $p \in \mathcal{P}^j$ gets corrupted, then all provers in \mathcal{P}^j get corrupted too.

iv. Upon receiving the $\text{CorruptParties}(Q)$ where $Q = \{q_1, \dots, q_{n'}\}$, the challenger runs:

- For a q_i that $q_i.\text{type} = \text{verifier}$, then $v.\text{Code} = q_i.\text{code}$ and $v.\text{Corr} = \text{true}$ for $v \in \mathcal{V}$.
- For each q_i that $q_i.\text{type} = \text{user}$ and $q_i.\text{usr} \leq m$, sets the users' revocation flag as $u_j.\text{Rev} = \text{true}$ where $j = q_i.\text{usr}$, runs $(\text{msk}', \text{gpk}') \leftarrow \text{Revoke}(\text{msk}, \text{gpk}, q_i.\text{usr})$, then update the group master key $\text{msk} \leftarrow \text{msk}'$ and the group public key $\text{gpk} \leftarrow \text{gpk}'$. This applies only if the AnonDB scheme has user revocation.
- For each q_i that $q_i.\text{type} = \text{prover}$, find the prover subset \mathcal{P}^j for $j = q_i.\text{usr}$. For each member p of subset \mathcal{P}^j , set their corruption flag $p.\text{Corr} = \text{true}$. If q_i is not corresponding to an existing prover, then create a new prover p and add it to the prover subset \mathcal{P}^j . Set the attributes of the participant p as follows:

a1. location $p.\text{Loc} = q_i.\text{location}$,

a2. execution code $p.\text{Code} = q_i.\text{code}$,

a3. start time $p.\text{St} = q_i.\text{time}$,

a4. corruption flag $p.\text{Corr} = \text{true}$, and

a5. secret-key $p.\text{Key} = u_j.\text{Cert}$.

- For each q_i that $q_i.\text{type} = \text{actor}$, add a new actor x to the set \mathcal{T} , and assign its attributes as follows:

a1. location $x.\text{Loc} = q_i.\text{location}$,

a2. execution code $x.\text{Code} = q_i.\text{code}$,

a3. start time $x.\text{St} = q_i.\text{time}$, and

a4. corruption flag $x.\text{Corr} = \text{true}$.

v. The challenger sends the key of the corrupted provers and the certificate of revoked

users to the adversary, i.e., $p.Key$ for all $p \in \mathcal{P}$ such that $p.Corr = true$ and $u.Cert$ for all $u \in \mathcal{U}$ such that $u.Rev = true$.

2. **Run:** Challenger activates all participants $x \in \mathcal{X} = \mathcal{P} \cup \mathcal{V} \cup \mathcal{T}$ at time $x.St$ for execution of $x.Code$.

The game ends when the last participant's code completes its execution.

We define five properties for anonymous distance-bounding protocols based on AnonDB Game, conditions to win the game varies from one property to another.

Property 5.2.1 (AnonDB Completeness). Consider an AnonDB scheme and an AnonDB game when $Q = \emptyset$ in the $\text{CorruptParties}(Q)$ query and the set \mathcal{P} is not empty.

The AnonDB scheme is (τ, δ) -complete for $0 \leq \tau, \delta \leq 1$, if the verifier returns $\text{Out}_V = 1$ with probability at least $1 - \delta$, under the following assumptions:

- the fast challenge-response rounds are independently affected by noise and at least τ portion of them are noiseless, and
- $\tau > 1 - p_{\text{noise}} - \epsilon$ for some constant $\epsilon > 0$.

A complete protocol must have negligible δ to be able to function in the presence of communication noises.

Property 5.2.2 (AnonDB Soundness) Consider an AnonDB scheme and an AnonDB game with the following restrictions:

- for all p in the nonempty set \mathcal{P} , and v as the only member of \mathcal{V} , we have $d(p.Loc, v.Loc) > \text{AnonDB}.\mathcal{B}$, and
- in the $\text{CorruptParties}(Q)$ query we have $q_i.type = actor$ for all $q_i \in Q$.

The AnonDB scheme is γ -sound if the probability of the verifier outputting $\text{Out}_V = 1$ is at most γ .

Lemma 18 A sound scheme according to Property 5.2.2 is resistant against relay attack [BC94],

mafia-fraud (Attack 3.1.2), impersonation attack (Attack 3.1.5), strong-impersonation [ASN17a], and collusion MF.

Proof 18 *The first four attacks are already shown to be included in soundness property in Lemma 3. Here we only show that for collusion MF.*

- *collusion MF: there is an honest verifier, multiple honest far-away provers, and a close-by MiM attacker who tries to convince the verifier that one of the provers is located close to the verifier. The attacker can have a learning phase before running the attack. The extra restrictions on the adversary in Property 5.2.2 is as follows:*
 - *the set of provers consists of a least two prover subsets, i.e., $\exists p_1, p_2 \in \mathcal{P} : p_1.\text{Key} \neq p_2.\text{Key}$, and*
 - *$\forall q_i \in Q$ we have $d(q_i.\text{location}, v.\text{Loc}) \leq \text{AnonDB}.\mathcal{B}$ for $v \in \mathcal{V}$.*

■

We consider two types of attacks by a dishonest prover: collusion far-away dishonest provers (Property 5.2.3), and collusion far-away dishonest provers with close-by helpers (Property 5.2.4).

Property 5.2.3 (AnonDB Distance-Fraud) *Consider an AnonDB scheme and an AnonDB game with the following restrictions:*

- *for all p in the nonempty set \mathcal{P} , and v as the only member of \mathcal{V} , we have $d(p.\text{Loc}, v.\text{Loc}) > \text{AnonDB}.\mathcal{B}$, and*
- *in the $\text{CorruptParties}(Q)$ query, $q_i.\text{type} = \text{prover}$ and $d(q_i.\text{location}, v.\text{Loc}) > \text{AnonDB}.\mathcal{B}$ for all $q_i \in Q$.*

The AnonDB scheme is α -DF-resistant if, for any AnonDB. Π protocol in such game, we have $\Pr[\text{Out}_V = 1] \leq \alpha$.

In the following we define the TF-resistance of anonymous DB protocols.

Property 5.2.4 (AnonDB Terrorist-Fraud) *Consider an AnonDB scheme and an AnonDB game*

with the following restrictions:

- for all p in the nonempty set \mathcal{P} , and v as the only member of \mathcal{V} , we have $d(p.Loc, v.Loc) > \text{AnonDB}.\mathcal{B}$,
- corrupted parties are either prover or actor, i.e., $\forall q_i \in \mathcal{Q} : q_i.type \in \{\text{prover}, \text{actor}\}$, and
- at least for one value of $j \in \{1 \dots m\}$ we have $d(q_i.location, v.Loc) > \text{AnonDB}.\mathcal{B}$ for all $q_i \in \mathcal{Q} \cap \mathcal{P}^j$.

The AnonDB scheme is μ -TF-resistant, if the following holds about the above game: if the verifier returns $\text{Out}_V = 1$ in the Π protocol of game Γ with non-negligible probability κ that is not traceable to any user with close-by provers (Property 5.2.6), then there is an impersonation attack (as an AnonDB game Γ' with honest verifier, no prover and one close-by actor) that takes the view of close-by participants of game Γ –excluding the verifier– as input, and makes the verifier return $\text{Out}_V = 1$ with probability at least $\kappa - \mu$ in the Π protocol of Γ' game.

In this definition, any directional message that is sent to the verifier from outside the distance bound, is not included in the input of the impersonator. Therefore any protocol that is secure in this property, is also secure against directional TF attacks. Note that this definition captures collusion TF (Figure 5.4 and Figure 5.5). In anonymous DB, breaking traceability is the only target of the adversary in collusion TF Type 2. Lemma 19 formally shows this claim.

The above attacks define security of the DB game. Now we define *anonymity* in terms of the distinguishing advantage of the adversary between two protocol sessions of two users.

Property 5.2.5 (AnonDB Anonymity) Consider an AnonDB scheme and an AnonDB game with the following restrictions:

- $\mathcal{P} = \{\mathcal{P}^1, \mathcal{P}^2\}$ where the size of each of the sets \mathcal{P}^1 and \mathcal{P}^2 is equal to $l > 0$, and
- in the $\text{CorruptParties}(\mathcal{Q})$ query, $q_i.type \in \{\text{verifier}, \text{actor}\}$ for all $q_i \in \mathcal{Q}$.

In this game, there are two subsets of honest provers of the same size, the adversary corrupts the verifier and adds a set of actors and sets their locations. Before activating the participants, the challenger randomly chooses $b \in_R \{0, 1\}^l$, and deactivates the i^{th} prover in $\mathcal{P}^{b[i]}$, i.e., $\forall 1 \leq i \leq l : \mathcal{P}_i^{b[i]}.Code = \emptyset$.

At the end of the game, \mathcal{A} returns $b' \in \{0, 1\}^l$. A protocol is α -anonymous if for any such experiment, for all values of $i \in \{1, \dots, l\}$ we have $|\Pr[b[i] = b'[i]] - \frac{1}{2}| \leq \alpha$.

We define *traceability* as a guarantee for the group manager to be able to identify the users from their protocol transcripts.

Property 5.2.6 (AnonDB Traceability) Consider an AnonDB scheme and an AnonDB game with the following restrictions:

- \mathcal{P} is nonempty, and
- in the $\text{CorruptParties}(Q)$ query, $q_i.type \in \{\text{prover}, \text{actor}\}$ for all $q_i \in Q$.

A protocol is called γ -traceable, if the success chance of the AnonDB.Open algorithm in identifying a user that has a prover in $\text{AnonDB}.\Pi$ protocol, from the transcript that is seen by the verifier, is at least as high as the chance of verifier outputting $\text{Out}_V = 1$ in the $\text{AnonDB}.\Pi$ protocol plus γ . i.e., $\Pr[\text{identify user}] \geq \gamma + \Pr[\text{Out}_V = 1]$.

Note that an AnonDB game considers multiple honest users being active at the same time. Therefore, all properties are according to collusion scenarios.

Lemma 19 An AnonDB scheme that is μ -TF-resistant according to Property 5.2.4, is μ' -directional TF-resistant for negligible values of μ and μ' .

Proof 19 The main difference between directional antenna and omni-directional antenna, from information security perspective, is that omni-directional antenna allows the participants, within the communication range, to have similar view from the transmitted messages, while the directional antenna makes the view of those participant to be different.

The rationality of Property 5.2.4 is that the higher the chance of future impersonation is, the scheme is more TF-resistant. So, the goal of a successful directional TF attack is to add the lowest amount of information to the view of the impersonation attacker.

In a TF attack (Property 5.2.4), all close-by participants, except the verifier, are controlled by the adversary. So, using any directional antenna to communicate with close-by participants that is not towards the verifier, is adding the transmitted message to the view of adversary. As a result, replacing that antenna with an omni-directional antenna does not reduce the knowledge of adversary, and so does not decrease its' chance in future impersonation. Therefore, we assume the only communications that are done by directional antenna, are those that are sent directly to the verifier.

Messages that are sent by directional antenna to the verifier, are not included in the view of impersonation adversary, i.e., View_β^Γ . Based on Property 5.2.4, if there is a TF attack against a scheme, the TF-resistant property guarantees the existence of impersonation attack by taking View_β^Γ as input, which is the minimum view of the adversary from a directional TF attack. Therefore, in a TF-resistant AnonDB scheme, a successful directional TF attack implies the existence of future impersonation attack. ■

5.3 AnonDB Construction: $\text{dbid2an}^{\text{GM}}$

We refer to our AnonDB scheme as $\text{dbid2an}^{\text{GM}}$ to emphasize conversion of a DBID scheme to an *anonymous* DBID. The DBID scheme has to use Goldwasser-Micali encryption system [GM84] for key generation. We first give an overview of our proposed scheme and then provide the details. In $\text{dbid2an}^{\text{GM}}$, a user is first enrolled in the system and is provided with a verifiable "membership" certificate. In addition to verifying the membership of a user, the certificate is used to generate a temporary public-key, which is later used in a public-key DBID protocol. At the end of a successful execution, the verifier is convinced that a valid member of the group is within the given distance

bound.

Recall (Definition 5.2.1) that for a security parameter λ , an anonymous distance-bounding (AnonDB) scheme is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathcal{B}, p_{\text{noise}}, \text{Init}, \text{CertGen}, \text{CertVer}, \Pi, \text{Revoke}, \text{Open})$. For our proposed AnonDB scheme, we denote these operations with $\text{dbid2an}^{\text{GM}}$ as the prefix of operation name, *i.e.*, $\text{dbid2an}^{\text{GM}}.\text{Init}$.

In $\text{dbid2an}^{\text{GM}}.\text{CertGen}$, the group manager generates a membership certificate for a new user, and accumulates the certificates of all users to form a public commitment on them. Then the $\text{dbid2an}^{\text{GM}}.\Pi$ protocol takes place as below:

- i) a prover of the user u_i , $i = 1..l$, anonymously proves that it owns one of the accumulated certificates (according to the public accumulated commitment).
- ii) a temporary public-key is generated for the prover. The temporary public-key is generated using Goldwasser-Micali encryption, *i.e.*, we have $C[j] = \text{Enc}^{\text{GM}}(x_l[j], v_l[j])$ where for the $j = 1 \dots \lambda$: $x_l[j]$ is certificate of the user, $v_l[j]$ is a random value chosen by the prover, and $C[j]$ is temporary public-key. In this paper we refer to $\text{Enc}^{\text{GM}}(.,.)$, as $\text{Commit}^{\text{GM}}(.,.)$ function. This temporary public-key generation is equivalent to the $\text{DBID}^{\text{GM}}.\text{KeyGen}$ function. After establishing the temporary public-key, the prover and the verifier run a $\text{DBID}^{\text{GM}}.\Pi$ protocol, where the prover uses (x_l, v_l) as input, and the verifier uses C as input.

In our construction of $\text{dbid2an}^{\text{GM}}$, we use ProProx [Vau14] as the DBID^{GM} scheme, which is proven secure in the model of DBID schemes (directional antenna and single user attacks) [ASN17a].

In this scheme we use a hash function H making randomness for $\text{Commit}^{\text{GM}}$, we define a deterministic commitment by

$$\text{Com}_{H_e}(x, v) := (cx_1, \dots, cx_\lambda, cv_1, \dots, cv_\lambda) \text{ for } x, v \in \mathbb{Z}_2^\lambda, \quad (5.3.1)$$

where $cx_j = \text{Commit}^{\text{GM}}(x_j, H(x, j).H(v, j)^e)$, $cv_j = \text{Commit}^{\text{GM}}(v_j, H(v, j))$, and $\text{Commit}^{\text{GM}}(.,.)$ being Goldwasser-Micali encryption function (Algorithm 2.6.2). We assume $H(0, i) = 1$ for all values

of $i \in \{1 \dots \lambda\}$, and also assume that Com_{H_e} is a one-way function (Same assumption is made in [Vau14, Section 4.1]).

Now we provide the details of the operations:

5.3.1 $(msk, gpk) \leftarrow \text{Init}(1^\lambda)$

The group manager initiates the system as follows:

- Initialize Goldwasser-Micali cryptosystem: $(p, q, N, \theta) \leftarrow \text{DBID}^{\text{GM}}.\text{Init}(1^\lambda)$ for λ bit security choose $N = p \cdot q$ for random prime values of p and q , and θ as a quadratic residue modulo N . Private: (p, q) and Public: (N, θ) .
- Initialize RSA cryptosystem for the same N : generate (d, e) such that $\gcd(e, \phi(N)) = 1$ and $d = e^{-1} \pmod{\phi(N)}$. d is private and e is public.

The group master key is $msk = (p, q, d, U)$ where U is the list of all user private-keys, initialized to $U = \emptyset$. The group public-key is $gpk = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$ where \hat{y} is commitment accumulation vector of user private-keys, \tilde{y} is signature vector of group manager on \hat{y} and Ξ is the list of all user membership signatures. These are initialized to $\hat{y} = \tilde{y} = [0]_\lambda$ and $\Xi = \emptyset$.

5.3.2 $(s, msk', gpk') \leftarrow \text{CertGen}(msk, gpk)$

The group manager first generates a certificate $s = (x_l, \sigma_l)$ and sends it to a new user (x_l is called user private-key, and σ_l is called user membership signatures). And second, the system master key and public-key get updated accordingly, i.e., $msk \leftarrow msk'$ and $gpk \leftarrow gpk'$. The details is as follows, assuming $l - 1$ users have already joined the group:

1. randomly choose $x_l \in \mathbb{Z}_2^\lambda$,
2. $y_l = Com_{H_e}(x_l, 0)$ using Equation 5.3.1.

3. $\forall j \in \{1, \dots, \lambda\}$:

- sign $\sigma_l[j] = (y_l[j])^d$,
- accumulate j^{th} bit of all user private-keys into a single bit $\hat{x}[j] = x_1[j] \oplus \dots \oplus x_l[j]$,
- accumulate hash values $\hat{v}[j] = \prod_{1 \leq i \leq l} H(x_i, j)$, and
- commit to accumulated values $\hat{y}[j] = \text{Commit}^{\text{GM}}(\hat{x}[j], \hat{v}[j]) = \theta^{\hat{x}[j]} \hat{v}[j]^2 \bmod N$.

4. Sign accumulated values $\tilde{y} = [\hat{y}[1]^{-d}, \dots, \hat{y}[\lambda]^{-d}]$.

The updated group master key is $msk' = (p, q, d, U)$ where $U = \{x_1, \dots, x_l\}$, and the updated group public-key is $gpk' = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$ where $\Xi = \{\sigma_1, \dots, \sigma_l\}$. The certificate $s = (x_l, \sigma_l)$ is securely sent to the new user.

5.3.3 $accept/reject \leftarrow \text{CertVer}(s, gpk)$

Upon receiving a certificate $s = (x, \sigma)$, the user can check its validity. By reading the group public-key $gpk = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$, the user calculates $y = \text{Com}_{H_e}(x, 0)$ and checks $y[j] \stackrel{?}{=} (\sigma[j])^e \bmod N$, for $j = \{1 \dots \lambda\}$.

5.3.4 $accept/reject \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$

When a prover (\mathcal{P}_l) of a registered user wants to run the AnonDB. Π protocol with the verifier, they will follow the protocol described in Figure 5.6. The protocol consists of two main steps. The first step is a message from the prover to the verifier (y', π) that generates a temporary public-key (C), and then provides a non-interactive zero-knowledge (NIZK), which proves that the prover knows the privates related to the temporary public-key C . Note that in the non-interactive zero-knowledge proof, the verifier does not send any message to the prover [BFM88, BG90]. The second step is running the DBID^{GM}. Π protocol.

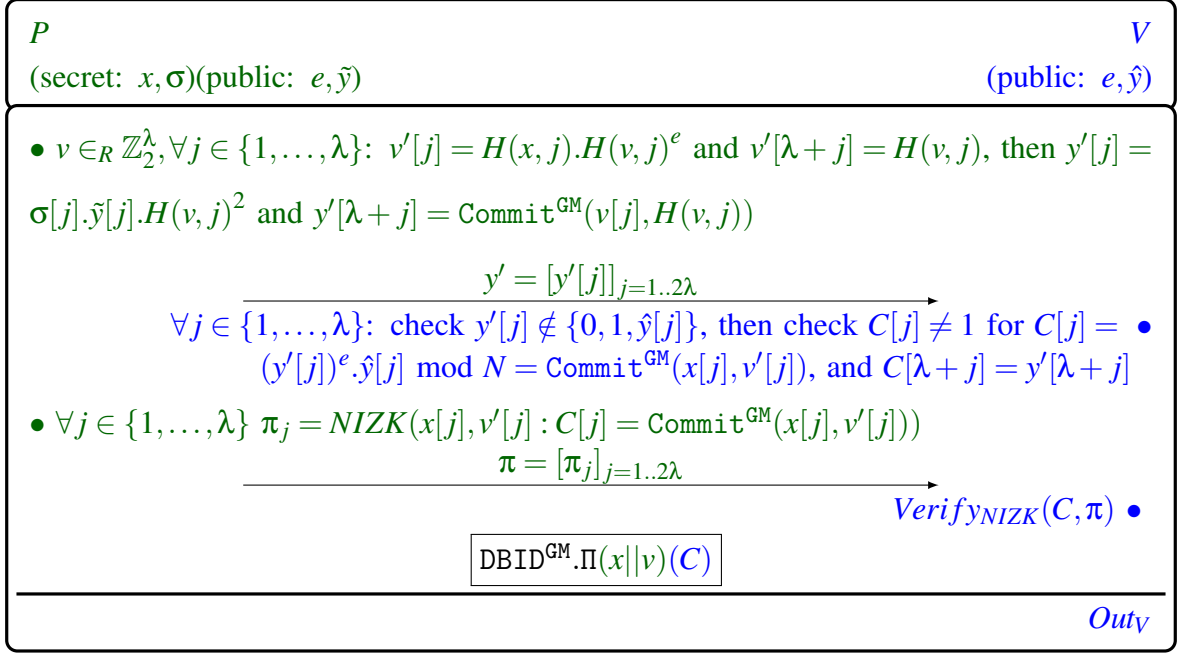


Figure 5.6: Π protocol in $\text{dbid2an}^{\text{GM}}$ scheme. $C = \text{Com}_{H_e}(x, v)$ using Equation 5.3.1.

5.3.5 $(I) \leftarrow \text{Open}(msk, \text{transcript})$

The tracing authority who holds the group master key msk , uses the verifier's view of a successful run of Π with the prover \mathcal{P}_I , and returns index of the corresponding user in \mathcal{U} . The algorithm runs as follows, knowing that the group master key is $msk = (p, q, d, U = \{x_1, \dots, x_m\})$:

1. $\hat{y}^d = [\hat{y}[1]^d, \dots, \hat{y}[\lambda]^d]$.
2. Parse verifier's view of the protocol to obtain y' and C .
3. Return the first $i \in \{1, \dots, m\}$ that all the following holds:
 - for all $j \in \{1, \dots, \lambda\}$ calculate v'_j from $v_j'^2 = y'[j] \cdot \hat{y}[j]^d \cdot (y_i[j])^{-d}$ (e.g. using Chinese Remainder Theorem, assuming that $v_j'^2$ is quadratic residue and the factorization of N is known),
 - then find $v[j] = H(x_i, j) \cdot v_j'^e$ or $v[j] = H(x_i, j) \cdot (-v_j')^e$, and

- check $C[j] \stackrel{?}{=} \text{Commit}^{\text{GM}}(x_i[j], v[j])$.

5.3.6 $(msk', gpk') \leftarrow \text{Revoke}(msk, gpk, l)$

In this operation, the entity holding the group master key msk , updates the group master key and the group public key such that the provers of l^{th} user ($l \in \{1 \dots m\}$) cannot succeed in any Π protocol anymore. The algorithm runs as follows, knowing that the group master key is $msk = (p, q, d, U = \{x_1, \dots, x_m\})$ and the group public key is $gpk = (e, N, \theta, \hat{y}, \tilde{y}, \Xi)$ where $\Xi = \{\sigma_1, \dots, \sigma_m\}$;

1. $\forall j \in \{1, \dots, \lambda\}$:

- $\hat{x}[j] = x_1[j] \oplus \dots x_{l-1}[j] \oplus x_{l+1}[j] \oplus \dots \oplus x_m[j]$,
- $\hat{v}[j] = \prod_{i \in \{1, \dots, l-1, l+1, \dots, m\}} H(x_i, j)$,
- $\hat{y}[j] = \text{Commit}^{\text{GM}}(\hat{x}[j]; \hat{v}[j]) = \theta^{\hat{x}[j]} \hat{v}[j]^2 \bmod N$,
- $\tilde{y}[j] = \hat{y}[j]^{-d}$, and

2. $\Xi' = \Xi \setminus \{\sigma_l\}$.

After this operation, the group master key is $msk' = (p, q, d, U = \{x_1, \dots, x_{l-1}, x_{l+1}, \dots, x_m\})$ and the group public key is $gpk' = (e, N, \hat{y}, \tilde{y}, \Xi')$.

5.3.7 Security Analysis

In this section we provide the security analysis of $\text{dbid2an}^{\text{GM}}$ protocol, assuming that the adopted DBID protocol is secure.

Theorem 3 ($\text{dbid2an}^{\text{GM}}$ **Security Properties**) *If (i) the DBID^{GM} scheme is (τ, δ) -complete, γ -sound, θ -DF-resistant, μ' -TF-resistant and $\text{DBID}^{\text{GM}}.\Pi$ is zero-knowledge, and (ii) the temporary public-key (C) and the private key (x_l, v_l) of $\text{DBID}^{\text{GM}}.\Pi$ are related as $C = \text{Enc}_N(x_l, v_l)$ where $\text{Enc}_N(.,.)$ is the Goldwasser-Micali encryption algorithm for modulus N with λ -bit security,*

then $\text{dbid2an}^{\text{GM}}$ is an AnonDB scheme that is (τ, δ) -complete (Property 5.2.1), θ -DF-resistant (Property 5.2.3), γ -Sound (Property 5.2.2), μ -TF-resistant (Property 5.2.4), α -anonymous (Property 5.2.5) and γ -traceable (Property 5.2.6), for negligible values of $\alpha, \delta, \gamma, \gamma', \mu, \mu'$ and θ , assuming that quadratic residuosity, factorization and RSA problems are hard problems.

In the following, we prove each of the properties of the theorem in a separate lemma. We prove security properties of the protocol based on the model described in Section 5.2. The underlying DBID^{GM} protocol provides single user directional antenna security [ASN17a]. The main challenge for the new model is to prove collusion security.

Lemma 20 (Completeness) $\text{dbid2an}^{\text{GM}}$ is a (τ, δ) -complete AnonDB (Property 5.2.1) scheme, if the DBID scheme is (τ, δ) -complete.

Proof 20 Consider an AnonDB game with $\text{dbid2an}^{\text{GM}}$ scheme, in which the provers and the verifier are honest. In each $\text{dbid2an}^{\text{GM}}.\Pi$ protocol, the steps before the DBID. Π protocol succeed. The DBID. Π protocol succeeds with probability at least δ , since the DBID scheme is (τ, δ) -complete. Therefore, the $\text{dbid2an}^{\text{GM}}.\Pi$ succeeds with probability at least δ , which implies (τ, δ) -completeness of $\text{dbid2an}^{\text{GM}}$ scheme. ■

Lemma 21 (DF-resistance) $\text{dbid2an}^{\text{GM}}$ is a θ -DF-resistant AnonDB (Property 5.2.3) scheme, if the DBID scheme is θ -DF-resistant.

Proof 21 In this proof, we reduce any successful AnonDB DF adversary to a successful DBID DF adversary. Consider an AnonDB game with $\text{dbid2an}^{\text{GM}}$ scheme, in which the provers are far-away and the verifier is honest. In each AnonDB. Π protocol, the verifier gets a temporary public-key and then participates in a DBID. Π protocol with that temporary public-key. Note that dishonest provers of a single user can operate simultaneously, that implies they can generate different temporary public-keys at the same time in different DBID. Π protocols. Therefore, having multiple users in the system does not increase the chance of adversary against the DBID scheme.

Let's consider the case that the verifier is simultaneously interacting with multiple provers (either

from one user or more) in different $\text{AnonDB}.\Pi$ protocols. We assume all the temporary public-key generations are successful. As a result, the verifier has access to a list of public-keys $\{C\}$, and provers have access to the corresponding secret-key x and the related randomness Δ . The relation between a public-key, secret-key and the corresponding randomness is $C = \text{Enc}(x||\Delta, r)$, where Enc is the Goldwasser-Micali encryption algorithm and r is pseudo-random.

After generation of the temporary public-keys, the adversary runs different $\text{DBID}.\Pi$ protocols simultaneously with the verifier. Since the DBID scheme is θ -DF-resistant, then for all instances of $\text{DBID}.\Pi$ protocols, the acceptance chance of the verifier is limited by negligible value θ . As a result, the acceptance chance of the $\text{dbid2an}^{\text{GM}}.\Pi$ protocol is limited by θ too. Note that the DF-resistant property of DBID scheme is considering collusion scenario. ■

Lemma 22 (TF-resistance) $\text{dbid2an}^{\text{GM}}$ is a μ -TF-resistant AnonDB (Property 5.2.4) scheme, if the DBID^{GM} scheme is μ' -TF-resistant for negligible values of μ and μ' .

Proof 22 According to the TF-resistance definition, we need to show that for $\text{dbid2an}^{\text{GM}}$ scheme, if there is a successful TF attack that is not traceable to any close-by prover, then one can impersonate the far-away prover with the view of all close-by participants, excluding the verifier. This is by assuming that the underlying $\text{DBID}^{\text{GM}}.\Pi$ scheme is single-user TF-resistant.

We divide the $\text{dbid2an}^{\text{GM}}.\Pi$ protocol into two parts: (i) temporary public-key generation, that is before the $\text{DBID}.\Pi$ protocol, and (ii) the $\text{DBID}^{\text{GM}}.\Pi$ protocol. In the first part, the verifier receives a message y' that allows it to calculate a temporary public-key C as a commitment on secret x and random v . And then in the second part, the two parties run the $\text{DBID}^{\text{GM}}.\Pi$ protocol based on the provided public-key C .

In any valid transcript that uses y' as the first commitment, the sub-transcript from the $\text{DBID}^{\text{GM}}.\Pi$ is a valid transcript according to the temporary public-key C , where for all values $j = 1..\lambda$ we have $C[j] = \text{Commit}^{\text{GM}}(x[j], H(x, j).H(v, j)^e)$ and $C[\lambda + j] = \text{Commit}^{\text{GM}}(v[j], H(v, j))$. Because of the binding property of commitment scheme $\text{Commit}^{\text{GM}}(.,.)$, finding any $x' \neq x$ such that $C[j] =$

$\text{Commit}^{\text{GM}}(x'[j], r)$ for all $j = 1.. \lambda$ is negligible, for any value of r . This implies finding any $x' \neq x$ such that $C = \text{Com}_{H_e}(x, v)$ is negligible. Therefore, succeeding in the $\text{DBID}^{\text{GM}}.\Pi$ sub-protocol with any prover input $x' \neq x_i$ is negligible.

Collusion TF: The only difference between a close-by prover of another user with a close-by actor, is the possession of secret value (x_i, σ_i) . The value of σ_i never gets used by the close-by prover, because it makes the session to be traceable to the close-by prover, which is not a TF attack based on definition. So we can consider the close-by prover owns the value x_i , while in a normal TF attack the close-by actor owns nothing (or a random value x'_i). Since x_i is randomly chosen by the group manager in CertGen operation, the statistical difference between x_i and x is the same as the statistical difference between x'_i and x . Therefore, possession of x_i or x'_i by a close-by party in helping the TF attack against $\text{DBID}^{\text{GM}}.\Pi$ sub-protocol with public-key C , makes no difference in success chance of the attack. So we can replace the close-by prover of another user with an actor. Let's consider a successful TF attack \mathcal{J} that succeeds with non-negligible probability κ . If the transcript is traceable to a close-by prover, this is not an attack according to the definition. Now we consider success chance of attack when no close-by prover is traceable. Since \mathcal{J} generates a transcript that is valid with probability κ , then the sub-transcript from $\text{DBID}^{\text{GM}}.\Pi$ is valid with at least probability κ . And according to the TF-resistance property of $\text{DBID}^{\text{GM}}.\Pi$, there is an impersonator \mathcal{E}_{dbid} for the $\text{DBID}^{\text{GM}}.\Pi$ protocol that succeeds with probability $\kappa - \mu$ for negligible μ . \mathcal{E}_{dbid} takes the view of all close-by participants in the attack \mathcal{J} , excluding the verifier, as input.

Impersonation: First we use \mathcal{E}_{dbid} to extract the secret x , then find the related σ as the first value in the public list Ξ that accept $\leftarrow \text{CertVer}(x, \sigma, \text{gpk})$. By having (x, σ) one can impersonate the prover.

Note that the key extraction x from impersonator \mathcal{E}_{dbid} , depends on the construction of $\text{DBID}^{\text{GM}}.\Pi$ protocol. In zero-knowledge based models, such as ProProx [Vau14], the impersonator \mathcal{E}_{dbid} extracts the key x itself. However, in identification based models, \mathcal{E}_{dbid} generates a valid Σ -protocol

transcript, i.e., $(A, [c], [r])$ for random $[c]$. Here we use the following technique to extract the key:

We divide \mathcal{E}_{dbid} into two parts: \mathcal{J}_1 is from the beginning of attack up to after the verifier receives A , and \mathcal{J}_2 is after that till the end. The first part \mathcal{J}_1 is run independent from the verifier, and the challenge values $[c]$ are randomly chosen by the verifier, where $[c]$ is n bits.

Key extractor: Run \mathcal{J}_1 once, followed by polynomial ℓ times of \mathcal{J}_2 . Before running \mathcal{J}_2 at any time, we rewind the memory state of the algorithm to the end of \mathcal{J}_1 . This generates the set Σ with ℓ transcripts (each valid with probability $\kappa - \mu$), where $[c]$ is randomly generated for each of them. If ℓ is chosen polynomially large enough in n , then for every fast-phase challenge-response bit of $[c]$, there are at least two valid transcripts that have different values on that bit. An index i is called bad index, if no pair of transcripts in Σ have complementary values in this challenge index, which happens with negligible probability $2^{-\ell}$. This allows us to extract the whole fast-phase challenge-response table with probability at least $\kappa - \mu - \lambda \cdot 2^{-\ell}$, where $\lambda \leq n$ is the length of the table (and size of key x). Finally by having the table, we can find the key x . ■

Lemma 23 (Soundness) $\text{dbid2an}^{\text{GM}}$ is a γ -sound AnonDB (Property 5.2.2) scheme for $\gamma = \text{negl}(\lambda)$ if $\text{DBID}.\Pi$ is $\text{negl}(\lambda)$ -sound and zero-knowledge, assuming that quadratic residuosity, factorization and RSA problems are hard problems.

Proof 23 Before starting the proof, we need to note that since the $\text{dbid2an}^{\text{GM}}$ scheme does not have user revocation, then the corruption query of adversary only consists of actors. i.e., there is no user in the corruption query.

According to the AnonDB game settings, we have some prover subsets $\mathcal{P}^j \in \mathcal{P}$ that there is no overlap in the execution time of any list \mathcal{P}^j , however the provers of two different subsets $\mathcal{P}^j \neq \mathcal{P}^i$ can run simultaneously. The corrupted actors of \mathcal{T} are controlled by the adversary.

In this game, the adversary succeeds if among all the successful AnonDB. Π protocols (Π^{succ} set), at least one of the following conditions hold:

- (i) $\exists \pi \in \Pi^{succ}, \forall p \in \mathcal{P} : t = fshTime(\pi), t \notin [p.St, p.St + exLen(p.Code)]$
- (ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{succ}, v \in \mathcal{V} : t = fshTime(\pi), t \in [p.St, p.St + exLen(p.Code)] \wedge d(p.Loc, v.Loc) > AnonDB.B.$

In this proof, we calculate the success chance of the adversary in both conditions.

First we specify the view and effect of the adversary; as a Σ^ -protocol (Definition 2.4.4), the honest parties expect three types of messages in the following order: (1) commitment A , (2) challenge sequence $[c]$, and (3) response sequence $[r]$. In protocol $dbid2an^{GM}.\Pi$ each of these messages are as follows:*

- (1) $A = DBID.\Pi.A$,
- (2) $[c] = DBID.\Pi.c$, and
- (3) $[r] = (DBID.\Pi.r, y', \pi)$.

Based on the definition, the adversary is able to modify or generate any of these messages. Now we consider the two winning conditions of the adversary:

Win Condition (i): No prover. *The first condition for the adversary is equivalent to generating a valid transcript $(A, [c], [r])$ with random challenges $([c])$, without the help of any prover. In order to succeed, \mathcal{A} needs to successfully pass the $DBID.\Pi$ protocol, i.e., generate a valid transcript $(DBID.\Pi.A, DBID.\Pi.c, DBID.\Pi.r)$ for public-key $C[j] = (y'[j])^e \cdot \hat{y}[j] \bmod N$ and $C[j + \lambda] = y'[j + \lambda]$ for $j = 1 \dots \lambda$, where y' is included in $[r]$.*

\mathcal{A} has to choose the value of y' in a way that the components of derived temporary public-key $C[j] = Commit^{GM}(X, Z)$ for $j = 1 \dots \lambda$ is known to the them, as otherwise the success chance of generating a valid ZKP $\pi \in [r]$ is negligible. Therefore, \mathcal{A} needs to find a tuple $(x = X, y' = Y, \Delta = Z)$ such that it holds in the following relation with the public parameters: $Commit^{GM}(X[j], Z[j]) = (Y[j])^e \cdot \hat{y}[j] \bmod N$ for $j = 1 \dots \lambda$. In order to succeed, \mathcal{A} needs to solve at least one of the following

problems; (a) find some information about the secret of a registered user, or (b) forge the tuple (X, Y, Z) such that X is independent from the secret of any registered user.

Case (i.a) We want to find the probability of any information leakage in $\text{dbid2an}^{\text{GM}}.\Pi$ experiment. The provers sends three pieces of information that is dependent to the secret: $\text{DBID}.\Pi.r$, y' , and π . The message π is zero-knowledge with independent randomness by definition and same about the $\text{DBID}.\Pi$ protocol according to the assumptions, so we can remove them from the view of \mathcal{A} .

As a result, we only need to find the probability of information leakage in the message y' . Since the message y' perfectly pads the private certificate values with fresh randomness, so the collection of multiple messages of y' does not help the adversary to leak any information about the secrets (x, σ) . Therefore the adversary is limited to break the computational hiding of $\text{Commit}^{\text{GM}}(X, Y) = \theta^X Y^2 \bmod N$ function in order to find the committed values. Note that each bit of the secrets (x, σ) are independent in the protocol, so \mathcal{A} 's chance in gaining any information about the secrets is negligible.

Case (i.b) \mathcal{A} has to find a tuple (X, Y, Z) such that $Y[j]^e = \frac{\text{Commit}^{\text{GM}}(X[j], Z[j])}{\hat{y}_j} = \frac{\theta^{X[j].Z[j]^2}}{\hat{y}_j}$ for $j = 1 \dots \lambda$. Note that the adversary have seen many values of Y in the learning phase, without knowing the related values of X and Z . Moreover, the learning phase values of Y look random to the adversary as they are perfectly padded by fresh randomness. So we can remove them from the view of adversary, which makes the view of adversary to be completely random (i.e., $\text{View}_{\mathcal{A}} = \emptyset$).

As a result, in order to find this tuple, \mathcal{A} has to solve this equation that needs solving at least one of the following three hard problems:

- Finding $Y[j]$ as e^{th} root of $\frac{\theta^{X[j].Z[j]^2}}{\hat{y}_j}$.
- Finding $Z[j]$ as square root of $\frac{\hat{y}_j.Y[j]^e}{\theta^{X[j]}}$.

- Finding $X[j]$ as discrete log of $\frac{\hat{y}[j].Y[j]^e}{Z[j]^2}$.

Therefore, all possible ways of soundness adversary to succeed under the condition (i) have negligible chance.

Win Condition (ii): Far-away provers. In the following we consider the condition (ii) by assuming that the adversary has no information about the secret of any of the provers involved in the AnonDB game. Without loss of generality, we assume there are only two active provers with two different secrets $((x_1, \sigma_1)$ and (x_2, \bar{y}_2)). Since the provers are honest, then they generate two independent values for y'_1 and y'_2 as each one is padded with fresh randomness. Let's assume there is a MiM adversarial algorithm \mathcal{A} , in which the provers have (x_1, Δ_1) and (x_2, Δ_2) as their secret in the DBID. Π protocols and the verifier accepts with non-negligible probability, while C is the temporary public-key that the verifier calculates. Here we consider two cases; (a) there exists $b \in \{1, 2\}$ that $C = \text{Com}_{H_e}(x_b, \Delta_b)$, (b) otherwise:

Case (ii.a) Without loss of generality, we assume $C = \text{Com}_{H_e}(x_1, \Delta_1)$. Now let's consider the DBID. Π sub-protocol in this setting. We name the related sub-procedure of \mathcal{A} that runs during the DBID. Π protocol, as $\mathcal{A}^{\text{DBID}}$. Since $\text{dbid2an}^{\text{GM}}.\Pi$ includes the DBID. Π protocol, then the acceptance of the verifier in a $\text{dbid2an}^{\text{GM}}.\Pi$ session implies the acceptance of the DBID. Π sub-protocol. Therefore, the $\mathcal{A}^{\text{DBID}}$ algorithm is a successful MiM adversary for the DBID protocol with non-negligible success chance. This is in contradiction with the $\text{negl}(\lambda)$ -soundness property of the DBID protocol.

Case (ii.b) Since both of the active provers generate non-interactive-ZKP for a different public-key value than C , then the adversary cannot send those proofs to the verifier, because both $\text{Verify}(C, \pi_1)$ and $\text{Verify}(C, \pi_2)$ fail. Therefore, the adversary has to generate a different π such that $\text{Verify}(C, \pi)$ succeeds. This implies that the related secret (x, Δ) is different from the secrets of the two far-away provers. As a result, in the sub-experiment of the DBID. Π protocol, the two far-away provers are counted as actors. Therefore, any non-negligible success chance

in the DBID. Π protocol is in contradiction with the $\text{negl}(\lambda)$ -soundness property of the DBID protocol.

Therefore, all possible ways of soundness adversary to succeed under the condition (ii) have negligible chance. ■

In above attacks, security against collusion attacks is obtained by simulating the credentials of extra users without having considerable impact on the success chance of the attacker, hence reducing the security to the case of single-user security model. To capture directional TF attack, we reduce the view of the impersonator messages that are sent directly to the helper.

Lemma 24 (Anonymity) $\text{dbid2an}^{\text{GM}}$ is an α -anonymous AnonDB (Property 5.2.5) scheme for $\alpha = \text{negl}(\lambda)$, if the DBID $^{\text{GM}}$ scheme is zero-knowledge.

Proof 24 We consider users $\mathcal{U} = \{u_1, u_2\}$ where $u_b = (x_b, \sigma_b)$ for $b \in \{1, 2\}$, and two prover subsets of the same size (i.e., $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2$ and $|\mathcal{P}^1| = |\mathcal{P}^2| = n$). There is no overlap in the execution time of any prover subset \mathcal{P}^j , however the provers of two different subsets $\mathcal{P}^j \neq \mathcal{P}^i$ can run simultaneously. The corrupted actors \mathcal{T} and the verifier \mathcal{V} are controlled by the adversary. The view of the adversary at the end of this game is: $\forall i \in \{1, \dots, n\}, b_i \in_R \{1, 2\} : (y'_{b_i}, \pi_i, \text{View}_{\mathcal{A}}^{\text{DBID}^{\text{GM}}.\Pi})$.

The values π_i and $\text{View}_{\mathcal{A}}^{\text{DBID}^{\text{GM}}.\Pi}$ are the outputs of the two zero-knowledge protocols. Therefore, there is an efficient simulator \mathcal{S} that can simulate both of these values without having access to the secrets (x_{b_i}, v_i) , without decreasing distinguishing advantage of adversary by a non-negligible amount. We thus consider the simulated view of adversary as: $b_i \in_R \{1, 2\} : y'_{b_i}$ for $i = 1 \dots n$. However, Since each element of y'_{b_i} is padded with a fresh pseudo-random (i.e., padded with $H(x, j) \cdot H(v, j)^e$ for $1 \leq j \leq \lambda$ and $H(v, j)$ for $\lambda < j \leq 2\lambda$, where v is random), the simulated view of the adversary computationally looks random (i.e., $\text{View}_{\mathcal{A}} = \emptyset$) and guessing b_i will remain random. ■

Lemma 25 (Traceability) $\text{dbid2an}^{\text{GM}}$ is a γ -traceable AnonDB (Property 5.2.6) scheme for $\gamma = \text{negl}(\lambda)$.

Proof 25 Consider an AnonDB game with $\text{dbid2an}^{\text{GM}}$ scheme, in which the verifier are honest. In each $\text{dbid2an}^{\text{GM}}.\Pi$ protocol that the verifier accepts, the Open algorithm identifies the user, unless the prover doesn't use the certificate of a user (i.e., forgery), which has negligible probability according on soundness property (Lemma 23). So we have $\Pr[\text{identify user}] \geq \gamma + \Pr[\Pi \text{ succeeds}]$, which implies $\text{negl}(\lambda)$ -traceability. ■

5.4 AnonDB Construction: $\text{dbid2an}^{\text{P}}$

In this section we show how the approach in Section 5.3 can be used on a public-key protocol with a different cryptosystem to construct another AnonDB scheme.

We refer to our AnonDB scheme as $\text{dbid2an}^{\text{P}}$ to emphasize conversion of a DBID scheme to an *anonymous* DBID. The DBID scheme has to use Pedersen commitment scheme [Ped92] for key generation. We first give an overview of our proposed scheme and then provide the details. In $\text{dbid2an}^{\text{GM}}$, a user is first enrolled in the system and is provided with a verifiable "membership" certificate. The membership certificate is generated by a CLSig signature scheme (Definition 2.6.4), such as BBS^+ [CL04].

In addition to verifying the membership of a user, the certificate is used to generate a temporary public-key, which is later used in a public-key DBID protocol. At the end of a successful execution, the verifier is convinced that a valid member of the group is within the given distance bound.

Recall (Definition 5.2.1) that for a security parameter λ , an anonymous distance-bounding (AnonDB) scheme is defined by a tuple $(\mathbb{X}, \mathbb{Y}, \mathbb{S}, \mathbb{B}, p_{\text{noise}}, \text{Init}, \text{CertGen}, \text{CertVer}, \Pi, \text{Revoke}, \text{Open})$. For our proposed AnonDB scheme, we denote these operations with $\text{dbid2an}^{\text{P}}$ as the prefix of operation name, i.e., $\text{dbid2an}^{\text{P}}.\text{Init}$. Note that $\text{dbid2an}^{\text{P}}$ scheme does not have Open and Revoke operations.

In $\text{dbid2an}^{\text{P}}.\text{CertGen}$, the group manager generates a membership certificate for a new user, by

running the BSign protocol of CLSig signature scheme. Then $\text{dbid2an}^P.\Pi$ protocol takes place as below:

- (i) a prover of the user u_i , $i = 1..l$, anonymously proves that it owns a membership certificate signed by group manager, by running the SPK protocol of CLSig signature scheme.
- (ii) a temporary public-key is generated for the prover. The temporary public-key is generated by using Pedersen commitment, as a result of SPK protocol. So we have $C = \text{Commit}^P(x, \Delta)$ where x is secret of the user, Δ is a random value chosen by the prover, and C is temporary public-key. This temporary public-key generation is equivalent to the $\text{DBID}^P.\text{KeyGen}$ function. After establishing the temporary public-key, the prover and the verifier run a $\text{DBID}^P.\Pi$ protocol, where the prover uses (x, Δ) as input, and the verifier uses C as input.

In our construction of dbid2an^P , we use POXY (Section 4.3) as the DBID^P scheme, which is proven secure in the model of DBID schemes (directional antenna and single user attacks).

Now we provide the details of the operations:

5.4.1 $(msk, gpk) \leftarrow \text{Init}(1^\lambda)$

The group manager initiates the system as follows:

- Initialize Pedersen commitment: $(msk, pk, p) \leftarrow \text{CLSig.KeyGen}(1^\lambda)$ for λ bit security choose large prime p . Private: (msk) and Public: (pk, p) .

The group master key is msk ; the group public-key is $gpk = (pk, p, \Xi)$, where Ξ is the list of all user membership signatures that is initialized to $\Xi = \emptyset$.

5.4.2 $(s, msk', gpk') \leftarrow \text{CertGen}(msk, gpk)$

The group manager generates a membership certificate $(s = (x, \sigma))$ and sends securely to the new user. The public parameters of the system are updated accordingly, i.e., $msk \leftarrow msk'$ and $gpk \leftarrow$

gpk' . The details are as follows:

1. randomly chooses $x \in \text{CLSig}.\mathbb{M}$, and
2. sign x using $\sigma \leftarrow \text{CLSig}.\text{Sign}(x, msk)$.

The group master key stays unchanged and is $msk' = msk$. The updated group public-key is $gpk' = (pk, p, \Xi')$ for $\Xi' = \Xi \cup \sigma$. The certificate $s = (x, \sigma)$ is securely sent to the new user.

This operation can also be implemented as a protocol between the user and group manager, *i.e.*, $\text{CertGen}\{U(gpk) \leftrightarrow GM(msk, gpk)\}$. The steps of protocol would be as follows

1. U randomly chooses $x \in \text{CLSig}.\mathbb{M}$, and
2. U and GM run the blind signature protocol $\text{CLSig}.\text{BSign}\{U(x, pk, p) \leftrightarrow GM(msk)\}$.

At the end of this protocol, both the user and the group manager output a signature σ on the message x .

5.4.3 $accept/reject \leftarrow \text{CertVer}(s, gpk)$

Upon receiving a certificate $s = (x_l, \sigma_l)$, the user can check its validity. By reading the group public-key $gpk = (pk, p, \Xi)$ for $\Xi = \{\sigma_1, \dots, \sigma_l\}$, the user checks if σ is included in Ξ and verifies its' validity using $accept \leftarrow \text{CLSig}.\text{Verify}(x, \sigma, pk, p)$ function.

5.4.4 $accept/reject \leftarrow \Pi\{P(s, gpk) \leftrightarrow V(gpk)\}$

When a prover (\mathcal{P}_l) of a registered user wants to run the $\text{AnonDB}.\Pi$ protocol with the verifier, they will follow the protocol described in Figure 5.7. The protocol consists of two main steps. The first step is a message from the prover to the verifier (π) that includes a temporary public-key C on prover's secret x and a non-interactive $\text{CLSig}.\text{SPK}$ which proves that the prover knows a signature of the group manager on the secret x without leaking information about the secret or the signature. The second step is running the $\text{DBID}^P.\Pi$ protocol.

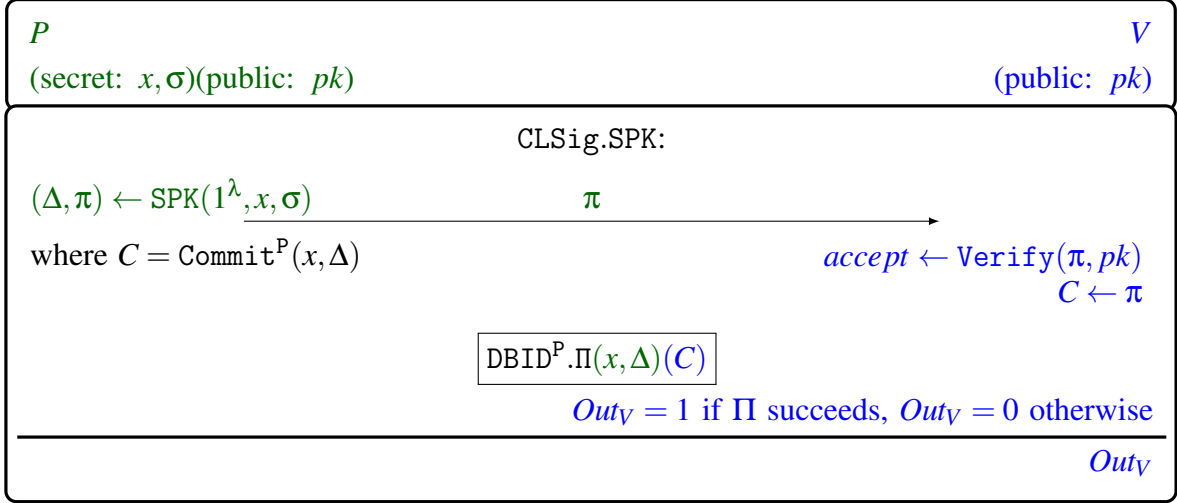


Figure 5.7: Π protocol in dbid2an^P scheme for the l^{th} user. $\text{Commit}^P(x, \Delta)$ is the Pedersen commitment function as $\text{Commit}^P(x, \Delta) = g^x h^\Delta \bmod p$ (Algorithm 2.6.1). Note that we are using a non-interactive protocol CLSig.SPK, which allows us to break down the protocol into two pieces $\text{CLSig.SPK} = (\text{SPK}, \text{Verify})$. Note that the value of C is embedded inside π , and we use the notation $C \leftarrow \pi$ to indicate the extraction of C from π . An instance of $\text{DBID}^P.\Pi$ sub-protocol is shown in Figure 4.5.

5.4.5 Security Analysis

In this section we provide the security analysis of dbid2an^P protocol, assuming that the adopted DBID protocol and CLSig schemes are secure.

Theorem 4 (dbid2an^P **Security Properties**) *If (i) the DBID^P scheme is (τ, δ) -complete, γ' -sound, θ -DF-resistant, μ' -TF-resistant and $\text{DBID}^P.\Pi$ is zero-knowledge protocol, (ii) CLSig scheme is complete, unforgeable, sound, and zero-knowledge with non-interactive SPK protocol based on the CLSig model (Definition 2.6.4), and (iii) the temporary public-key C and the secret key x of $\text{DBID}^P.\Pi$ are related as $C = \text{Commit}^P(x, \Delta)$ for a known value of Δ to prover, where $\text{Commit}^P(.,.)$ is Pedersen commitment,*

then dbid2an^P is an AnonDB scheme that is (τ, δ) -complete (Property 5.2.1), θ -DF-resistant (Property 5.2.3), γ -Sound (Property 5.2.2), μ -TF-resistant (Property 5.2.4), and α -anonymous (Property 5.2.5), for negligible values of $\alpha, \delta, \gamma, \gamma', \mu, \mu'$ and θ , assuming that the discrete logarithm is a hard problem.

In the following, we prove each of the properties of the theorem in a separate lemma. We prove security properties of the protocol based on the model described in Section 5.2. The underlying DBID^P protocol provides single user directional antenna security. The main challenge for the new model is to prove collusion security.

Lemma 26 (Completeness) dbid2an^P is a (τ, δ) -complete AnonDB (Property 5.2.1) scheme, if the CLSig scheme is complete and the DBID scheme is (τ, δ) -complete.

Proof 26 Consider an AnonDB game with dbid2an^P scheme, in which the provers and the verifier are honest. In each $\text{dbid2an}^P.\Pi$ protocol, since the CLSig scheme is complete, then the verifier accepts in the CLSig.SPK protocol and receives the correct value of C . Since the DBID scheme is (τ, δ) -complete, then the verifier accepts the DBID. Π protocol with probability at least δ . Therefore, the dbid2an^P scheme is (τ, δ) -complete. ■

Lemma 27 (DF-resistance) dbid2an^P is a θ -DF-resistant AnonDB (Property 5.2.3) scheme, if the DBID scheme is θ -DF-resistant.

Proof 27 In this proof, we reduce any successful AnonDB DF adversary to a successful DBID DF adversary. Consider an AnonDB game with dbid2an^P scheme, in which the provers are far-away and the verifier is honest. In each AnonDB. Π protocol, the verifier gets a commitment and then participates in a DBID. Π protocol with that commitment. Note that dishonest provers of a single user can operate simultaneously, that implies they can generate different commitments at the same time in different DBID. Π protocols. Therefore, having multiple users in the system does not increase the chance of adversary.

Let's consider the case that the verifier is simultaneously interacting with multiple provers (either

from one user or more) in different $\text{AnonDB}.\Pi$ protocols. We assume all the commitment generations are successful. As a result, the verifier has access to a list of commitments $\{C\}$, and the provers have access to the corresponding secret-key x and the related randomness Δ . The relation between a commitment, secret-key and the corresponding randomness is $C = \text{Commit}^P(x, \Delta)$, where Commit^P is the Pedersen commitment.

After generation of the commitment, the adversary runs different $\text{DBID}.\Pi$ protocols simultaneously with the verifier. Since the DBID scheme is θ -DF-resistant, then for all instances of $\text{DBID}.\Pi$ protocols, the acceptance chance of the verifier is limited by negligible value θ . As a result, the acceptance chance of the $\text{dbid2an}^P.\Pi$ protocol is bounded by θ too. Note that the DF-resistant property of DBID scheme is considering collusion scenario. ■

Lemma 28 (TF-resistance) dbid2an^P is a μ -TF-resistant AnonDB (Property 5.2.4) scheme, when there is no close-by prover, if the DBID scheme is μ' -TF-resistant for negligible values of μ and μ' .

Note that since there is no traceability in dbid2an^P scheme, then the case of having the special case of TF attack as an AnonDB game (Property 5.2.4) with far-away provers of user u_1 and close-by provers of user u_2 is meaningless. Therefore, as stated in this lemma, we do not consider any close-by prover, which means the only close-by participants are actors and the verifier.

Proof 28 According to the TF-resistance definition, we need to show that for dbid2an^P scheme, if there is a successful TF attack, then one can impersonate the far-away prover with the view of all close-by participants, excluding the verifier. This is by assuming that the underlying $\text{DBID}^P.\Pi$ scheme is single-user TF-resistant.

We divide the $\text{dbid2an}^P.\Pi$ protocol into two parts: (i) temporary public-key generation that is by CLSig.SPK protocol, and (ii) the $\text{DBID}.\Pi$ protocol. In the first part, the verifier receives a message π that allows it to extract a temporary public-key C as a commitment on secret x . And then in the second part, the two parties run the $\text{DBID}^P.\Pi$ protocol based on the provided public-key C .

In any valid transcript that uses π as the first commitment, the sub-transcript from the $\text{DBID}^P.\Pi$

is a valid transcript according to the temporary public-key C , where $C = \text{Commit}^P(x, \Delta)$. Note that because of the binding property of commitment scheme $\text{Commit}^P(,)$, finding any $x' \neq x$ such that $C = \text{Commit}^P(x', r)$ is negligible, for any value of r . Therefore, succeeding in the $\text{DBID}^P.\Pi$ sub-protocol with any prover input $x' \neq x$ is negligible.

Let's consider a successful TF attack \mathcal{J} that succeeds with non-negligible probability κ . Since \mathcal{J} generates a transcript that is valid with probability κ , then the sub-transcript from $\text{DBID}^P.\Pi$ is valid with at least probability κ . And according to the TF-resistance property of $\text{DBID}^P.\Pi$, there is an impersonator \mathcal{E}_{dbid} for the $\text{DBID}^P.\Pi$ protocol that succeeds with probability $\kappa - \mu$ for negligible μ . \mathcal{E}_{dbid} takes the view of all close-by participants in the attack \mathcal{J} , excluding the verifier, as input.

Impersonation: first we use \mathcal{E}_{dbid} to extract the secret x , then find the related σ as the first value in the public list Ξ that accept $\leftarrow \text{CertVer}(x, \sigma, \text{gpk})$. By having (x, σ) one can impersonate the prover.

We divide \mathcal{E}_{dbid} into two parts: \mathcal{J}_1 is from the beginning of attack up to after the verifier receives A , and \mathcal{J}_2 is after that till the end. The first part \mathcal{J}_1 is run independent from the verifier, and the challenge values $[c]$ are randomly chosen by the verifier, where $[c]$ is n bits.

Key extractor: run \mathcal{J}_1 once, followed by polynomial ℓ times of \mathcal{J}_2 . Before running \mathcal{J}_2 at any time, we rewind the memory state of the algorithm to the end of \mathcal{J}_1 . This generates the set Σ with ℓ transcripts (each valid with probability $\kappa - \mu$), where $[c]$ is randomly generated for each of them. If ℓ is chosen polynomially large enough in n , then for every fast-phase challenge-response bit of $[c]$, there are at least two valid transcripts that have different values on that bit. An index i is called bad index, if no pair of transcripts in Σ have complementary values in this challenge index, which happens with negligible probability $2^{-\ell}$. This allows us to extract the whole fast-phase challenge-response table with probability at least $\kappa - \mu - \lambda \cdot 2^{-\ell}$, where $\lambda \leq n$ is the length of the table (and size of key x). Finally by having the table, we can find the key x . ■

Lemma 29 (Soundness) dbid2an^P is a γ -sound AnonDB (Property 5.2.2) scheme for $\gamma = \text{negl}(\lambda)$

if (i) CLSig scheme is unforgeable, sound and zero-knowledge, and (ii) DBID scheme is $\text{negl}(\lambda)$ -sound and zero-knowledge, assuming that discrete logarithm is a hard problem.

Proof 29 Before starting the proof, we need to note that since the dbid2an^P scheme does not have user revocation, then the corruption query of adversary only consists of actors. i.e., there is no user in the corruption query.

According to Property 5.2.2, we have some prover subsets $\mathcal{P}^j \in \mathcal{P}$ that there is no overlap in the execution time of any list \mathcal{P}^j , however the provers of two different subsets $\mathcal{P}^j \neq \mathcal{P}^i$ can run simultaneously. The corrupted actors of \mathcal{T} are controlled by the adversary.

In this game, the adversary succeeds if among all the successful $\text{AnonDB}.\Pi$ protocols (Π^{succ} set), at least one of the following conditions hold:

- (i) $\exists \pi \in \Pi^{\text{succ}}, \forall p \in \mathcal{P} : t = \text{fshTime}(\pi), t \notin [p.\text{St}, p.\text{St} + \text{exLen}(p.\text{Code})]$
- (ii) $\exists p \in \mathcal{P}, \exists \pi \in \Pi^{\text{succ}}, v \in \mathcal{V} : t = \text{fshTime}(\pi), t \in [p.\text{St}, p.\text{St} + \text{exLen}(p.\text{Code})] \wedge d(p.\text{Loc}, v.\text{Loc}) > \text{AnonDB}.\mathcal{B}.$

In this proof, we calculate the success chance of the adversary in both conditions.

First we specify the view and effect of the adversary; as a Σ^* -protocol (Definition 2.4.4), the honest parties expect three types of messages in the following order: (1) commitment A , (2) challenge c , and (3) response r . In protocol $\text{dbid2an}^P.\Pi$ each of these messages are as follows:

- (1) $A = (\text{CLSig}.\text{SPK}.\pi^{\bar{x}}, \text{DBID}.\Pi.A)$, where $\text{CLSig}.\text{SPK}.\pi^{\bar{x}}$ is the sections of message π that are independent from x ,
- (2) $c = \text{DBID}.\Pi.c$, and
- (3) $r = (\text{CLSig}.\text{SPK}.\pi^x, \text{DBID}.\Pi.r)$, where $\text{CLSig}.\text{SPK}.\pi^x$ is the sections of message π that are dependent to x .

Based on the definition, the adversary is able to modify or generate any of these messages. Now

we consider the two winning conditions of the adversary:

Win Condition (i): No prover. *The first condition for the adversary is equivalent to generating a valid transcript (A, c, r) with random challenges (c) , without the help of any prover. In order to succeed, \mathcal{A} needs to successfully pass the DBID. Π protocol, i.e., generate a valid transcript $(\text{DBID}.\Pi.A, \text{DBID}.\Pi.c, \text{DBID}.\Pi.r)$ for a public-key C . Here we consider two cases about C : (a) $C = \text{Commit}^P(x, \Delta)$ where x is the secret of a user in set \mathcal{U} , (b) there is no user in \mathcal{U} that has the secret x , where $C = \text{Commit}^P(x, \Delta)$ and the adversary know the values of x and Δ .*

Case (i.a) *In order to succeed in the CLSig.SPK protocol, the adversary needs to either know x based on the soundness property of the CLSig scheme, or replay an earlier valid message π . Knowing x doesn't happen in this case, unless negligible probability, because it is in contradiction with the zero-knowledge property of the CLSig scheme and harness of discrete logarithm problem. Replaying the π message of a valid legitimate prover $p \in \mathcal{P}$, implies that the DBID. Π protocol is running with the same public-key C as the prover p has used in earlier DBID. Π protocol. This doesn't happen, unless negligible probability, because it is in contradiction with the soundness property of the DBID scheme.*

Case (i.b) *This case doesn't happen, unless negligible probability, because it is in contradiction with the soundness property of the CLSig scheme.*

Therefore, all possible ways of the MiM adversary to succeed under the condition (i) have negligible chance.

Win Condition (ii): Far-away provers. *In the following we consider the condition (ii) by assuming that the adversary has no information about the secret of any of the provers. Without loss of generality, we assume there are only two active provers with two different secrets (x_1, σ_1) and (x_2, σ_2) . Since the provers are honest, then they generate two different values of π_1 and π_2 that implies two public-keys $C_1 = \text{Commit}^P(x_1, r_1)$ and $C_2 = (x_2, r_2)$. The two values C_1 and C_2 are in-*

dependent, because the inputs of the related commitment functions are independent keys and fresh randomness.

Let's assume that there is a MiM adversarial algorithm \mathcal{A} , in which the provers have (x_1, r_1) and (x_2, r_2) as their secret in the DBID. Π protocols and the verifier accepts with non-negligible probability, while C is the temporary public-key that the verifier calculates. Here we consider two cases; (a) $C = \text{Commit}^P(x, \Delta)$ where $x \in \{x_1, x_2\}$, (b) there is no $x \in \{x_1, x_2\}$ where $C = \text{Commit}^P(x, \Delta)$ and the adversary know the values of x and Δ .

Case (ii.a) Without loss of generality, we assume $C = \text{Commit}^P(x_1, \Delta)$. Now let's consider the DBID. Π sub-protocol in this setting. We name the related sub-procedure of \mathcal{A} that runs during the DBID. Π protocol, as $\mathcal{A}^{\text{DBID}}$. Since $\text{dbid2an}^P.\Pi$ includes the DBID. Π protocol, then the acceptance of the verifier in a $\text{dbid2an}^P.\Pi$ session implies the acceptance of the DBID. Π sub-protocol. Therefore, the $\mathcal{A}^{\text{DBID}}$ algorithm is a successful MiM adversary for the DBID protocol with non-negligible success chance. This is in contradiction with the $\text{negl}(\lambda)$ -soundness property of the DBID protocol.

Case (ii.b) The active provers generate the messages π_1 and π_2 that respectively contain two independent public-keys C_1 and C_2 . Let's assume the adversary sends the message π to the verifier, that contains the public-key C . Based on the assumption of the case, the related x is not among $\{x_1, x_2\}$. Therefore, the adversary does not have access to a valid signature on x , based on the unforgeability property of the CLSig scheme. As a result, if the adversary succeed in the CLSig.SPK protocol with non-negligible probability, then we can use it to break the soundness property of the CLSig scheme.

Therefore, all possible ways of the MiM adversary to succeed under the condition (ii) have negligible chance. ■

Lemma 30 (Anonymity) dbid2an^P is an α -anonymous AnonDB (Property 5.2.5) scheme for $\alpha = \text{negl}(\lambda)$, if the CLSig scheme and the DBID scheme are zero-knowledge.

Proof 30 We consider users $\mathcal{U} = \{u_1, u_2\}$ where $u_b = (x_b, \sigma_b)$ for $b \in \{1, 2\}$, and two prover subsets of the same size (i.e., $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2$ and $|\mathcal{P}^1| = |\mathcal{P}^2| = n$). There is no overlap in the execution time of any prover subset \mathcal{P}^j , however the provers of two different subsets $\mathcal{P}^j \neq \mathcal{P}^i$ can run simultaneously. The corrupted actors \mathcal{T} and the verifier \mathcal{V} are controlled by the adversary. The view of the adversary at the end of this game is: $\forall i \in \{1, \dots, n\}, b_i \in_R \{1, 2\} : (\pi_i, \text{View}_{\mathcal{A}}^{\text{DBID}^{\mathcal{P}}.\Pi})$.

The values π_i and $\text{View}_{\mathcal{A}}^{\text{DBID}^{\mathcal{P}}.\Pi}$ are the outputs of two zero-knowledge protocols. Therefore, there is an efficient simulator \mathcal{S} that can simulate both of these values without having access to the secrets (x_{b_i}, Δ_i) , without decreasing distinguishing advantage of adversary by a non-negligible amount. Therefore, the simulated view of the adversary computationally looks random (i.e., $\text{View}_{\mathcal{A}} = \emptyset$) and guessing b_i will remain random. ■

5.5 Related Works

There are three known anonymous DB protocols [ASN14, BGG⁺16, ABG⁺17], that are designed to be secure against all distance-bounding attacks, which were all shown insecure against our proposed attacks.

[ASN14] formally defines *Distance-Fraud*, *Mafia-Fraud*, *Strong-Impersonation*, *Original Terrorist-Fraud*, *Distance-Hijacking* and considers *Anonymity* of provers. In this model, the verifier only has access to the public parameters of the system. However it has some disadvantages: the scheme does not provide revocation and uses the *Original TF* definition that is not appropriate for anonymous DB.

[BGG⁺16] proposed an anonymous distance bounding model, which considers *Distance-Fraud*, *Mafia-Fraud* and *Terrorist-Fraud* in addition to anonymity of provers against the verifier. This model achieves anonymity and revocability by using a revocable group signature scheme, that allows join, revocation and escrow operations for provers. However, in this protocol the verifier must be registered in the system which makes its application more limited compared to that of

[ASN14]. [ABG⁺17] uses the same model and structure as [BGG⁺16].

5.6 Concluding Remarks

We showed the security challenges that arise when identity information is not directly used in DB protocols, and proposed a new model that captures all known attacks and a construction with provable security in this model. We introduced two attacks; directional attack that uses the capability of an attacker at the physical layer of communication, and collusion attack that the provers of multiple user collude to deceive the verifier. And we showed that all existing anonymous DB schemes are vulnerable against our attack.

We proposed two constructions for different cryptosystems that convert public-key DB protocols to anonymous DB protocols. These constructions are modular and can use similar components that follows the designed cryptosystem. These two protocols are the first that are resistant against all distance-bounding attacks, including directional antenna attacks. The security properties of these protocols are provided.

Some parts of this chapter are published in [ASN14, ASNRA18].

Chapter 6

One-Shot Distance-Bounding

In this section we introduce One-Shot DB, as a one message authentication distance-bounding protocol, for the first time in the literature. In this model, we consider the concept of time and synchronization between parties that allows them to be able to verify the distance without using challenge-response. This technique reduces the fast phase messages from polynomial down to constant value, which makes the system very scalable. It also allows the parties to run the protocol while they are moving. Overall, this approach significantly widens the application of distance bounding protocols. We also propose a One-Shot DB construction and provide security proof and feasibility.

The goal of a distance bounding (DB) protocol is to provide assurance to an honest *verifier* that a party who is located within a distance bound \mathcal{B} from the verifier can prove their identity and that they are within the distance bound from the verifier, while a far-away party cannot. *One-shot DB* is a one-message protocol between a prover P with location loc_P and a secret key k_P , and a verifier V with location loc_V and matching k_V , where the message is sent from P to V , allowing V to verify, (i) P 's identity claim is valid, and (ii) $d(loc_P, loc_V) \leq \mathcal{B}$.

Traditional distance bounding protocols require multiple rounds of messages exchanged between prover and verifier in a time-sensitive manner, which is sometimes not desirable. For example, in some DB implementations it is suggested to use multiple “silent channels” for each time-sensitive message [RTŠ⁺12]. This leads to congestion and increased communication cost in populated area

with many provers. Reduction from multiple to single message can significantly improve this situation and can grant us scalability. For the same reason, one-shot DB protocols are also highly suitable in scenarios where prover needs to prove their proximity continuously over time. For example, in key-less entry car scenario, the user (prover) needs to be in the car all the time for the engine to work. One-shot DB protocols have execution lifespan much shorter than the traditional DB protocols, which makes them suitable for mobile prover scenarios, such as Mobile Ad-hoc Network (MANET) and Vehicular Ad-hoc Network (VANET).

Our work: This chapter describes one-message DB for the first time in the literature.. We propose a synchronized distributed model that entangles physical distance with time. We propose a construction that uses a global beacon system to orchestrate distributed parties.

Organization. Section 6.1 presents our model and in Section 6.2 we define typical distance-bounding attacks in the new model. In Section 6.3 we describe a class of One-Shot DB schemes, and in Section 6.4 we propose a specific construction for the scheme, and give security theorems and proofs. Section 6.5 concludes the chapter.

6.1 Model

We consider a system of multiple participants \mathcal{X} who can get involved in the protocol. A participant can either be a *party* or an *actor*. The adversary \mathcal{A} , which we describe later in this section, is not a participant.

Parties. \mathcal{P} denotes the set of n parties in the system, that can be either *prover* or *verifier*. Each party in \mathcal{P} has the following local variables: A local time T_i , a location loc_i and a long lived key k_i . The long-lived key is either a public key pair, or a list of $n - 1$ long-lived symmetric keys shared with the other parties. In this work we mainly consider symmetric keys.

A prover wants to prove to a verifier that they (i) have valid cryptographic authentication creden-

tials, and (ii) the distance between their location and the verifier's location is at most \mathcal{B} (they are *within distance* \mathcal{B}). *prover* can be corrupted by the adversary, but the *verifier* is honest.

Actors. \mathcal{T} denotes the set of actors with size n' , and each actor has following local variables: A local time T_i and a location loc_i . An actor does not have sufficient cryptographic key to allow them succeeding in an authentication claim. Adversary \mathcal{A} has access to the state of all the actors in the system, and can put algorithms/code of its choice inside any of them. Some of the actors are called *helpers*. A helper \mathcal{H} is associated with a prover p , and helps p to succeed in an authentication claim to a verifier v . A prover may have multiple helpers, and we assume that all the helpers of a prover are located within the distance bound \mathcal{B} to the verifier that this prover is trying to be authenticated to. So, the collusion between a prover who is not within the bound, and a helper within the bound, may result in a successful authentication claim.

All participants run local clock processes $clock_i()$, $i \in \{1, \dots, n' + n\}$ (e.g., internal computer clock) whose output increments T_i at regular intervals. Clock processes run at similar rate but may slightly drift over time - so although they can be initially synchronized, over time they can become out of synchronization.

6.1.1 Protocol

We adopt the notion of *protocol* from Bellare-Rogaway (Section 2.3 [BR93]) and extend it to model distance-bounding protocol. A protocol is a two party one, specified by a function Π on the following inputs:

- 1^λ — security parameter, $\lambda \in \mathbb{N}$
- i — identity of the sender, $i \in I \subseteq \{0, 1\}^\lambda$
- j — identity of the intended receiver, $j \in I \subseteq \{0, 1\}^\lambda$
- a — secret information of the sender, $a \in \{0, 1\}^*$

- $loc_{\mathcal{P}_i}$ — location of a participant \mathcal{P}_i
- $T_{\mathcal{P}_i}^0$ — initial value of the local clock of participant \mathcal{P}_i ,
- r — the random coin flips of the sender, $r \in \{0, 1\}^*$

The output value of $(m, \delta) \leftarrow \Pi(1^\lambda, i, j, a, loc_{\mathcal{P}_i}, T_{\mathcal{P}_i}^0, r)$ specifies:

- m — the next message to send out, $m \in \{0, 1\}^* \cup \{*\}$
- δ — the decision of the protocol, $\delta \in \{Acc, Rej, *\}$

I is a list of participants in the protocol. Although only two parties can participate in a protocol, the set of participants I could be larger. The value $m = *$ suggests that the participant sends no message. The values Acc and Rej for δ suggest *accept* and *reject*, respectively. The value $\delta = *$ suggests that the protocol has not reached a decision. Acceptance does not occur until the end of the protocol, while rejection may occur any time.

a is the secret information given to a prover. This is usually a long-lived key (LL-key) of a prover. In case of symmetric authentication, each pair of prover (P_i, P_j) will be given the same LL-key, and the adversary will be denied this key. We assume that a protocol is associated with a *long-lived key generator*, or LL-key generator $\mathcal{G}(1^\lambda, \mathfrak{t}, r_G)$. Here, the inputs are - the security parameter 1^λ , the identity of a party $\mathfrak{t}, \mathfrak{t} \in I \cup \{\mathcal{A}\}$, and an infinite string $r_G \in \{0, 1\}^*$, *i.e.*, coin flips of the LL-key generator. In this chapter, we consider the LL-key generator to be a symmetric one. For each pair of parties $\mathcal{P}_i, \mathcal{P}_j \in I$, we have $\mathcal{G}(1^\lambda, \mathcal{P}_i, r_G) = \mathcal{G}(1^\lambda, \mathcal{P}_j, r_G)$. We note that the adversary and the actors do not receive any key from the generator, that is, $\mathcal{G}(1^\lambda, \mathcal{A}, r_G) = \mathcal{G}(1^\lambda, \mathcal{T}, r_G) = \{0, 1\}^{\leq 0}$.

Each participant is given a location $loc_{\mathcal{P}_i} = (x, y) \in \mathbb{R} \times \mathbb{R}$, that is an element of a metric space equipped with Euclidean distance, and is fixed during the protocol. Distance between any two locations is returned by the distance function $d(loc_1, loc_2)$. Travel time of a message between locations loc_1 and loc_2 is $\frac{d(loc_1, loc_2)}{c}$, where c is the speed of light.

The participants that are located within a predefined distance bound \mathcal{B} from a verifier are called

close-by participants, and those who are outside the distance bound from the verifier are called *far-away* participants.

The local clock variable T_i of participant \mathcal{P}_i is initiated with the value $T_{\mathcal{P}_i}^0$. This is same for all the participants in a protocol.

6.1.2 Participants and Process Oracles

A participant $\mathcal{P}_i, i = 1 \cdots n$, possibly with key set $\{k_{i,j}, j = 1 \cdots n, j \neq i\}$, runs process oracles $\pi_{i,j}^s, s \in [1, \ell]$, where $\pi_{i,j}^s$ models participant \mathcal{P}_i attempting to authenticate itself to participant \mathcal{P}_j in its session s .

Process oracles and the participant itself are modeled as Turing machines, each with an input and output tape. Oracles can be *initialized*, can *send* and *receive* messages according to the protocol specification, and terminate either in *finished*, and *accept* or *reject* state. The complete set of states for oracles is $\Lambda = \{\text{not initialized, finished, accept, reject}\}$.

Long-lived keys of participant \mathcal{P}_i and their clock value t_i (read from variable T_i) are shared by all process oracles (or oracles, for short) $\pi_{i,j}^s, s \in [1, \ell]$ of this participant.

Computed nonces, intermediate state values, and session keys are only known to a single oracle. The current state of each oracle is stored in variable Λ and the transcript of all messages sent and received (in chronological order) in variable $T^{i,s}$.

Our security definitions consider one message protocols, and we need to distinguish between initiator and responding oracles.

Definition 6.1.1 *An oracle who sends the first message in a protocol is called initiator (prover), and an oracle who receives the first message is called responder (verifier).*

Local states of oracles. An initiator oracle $\pi_{i,j}^s$ will, after being initiated, *retrieves the actual time value t_i from its local clock variable*, prepare and send a message of the form $(P_i, P_j, t_i, m, \sigma)$ where P_j denotes the identity of the intended receiving party, t_i is the local time of P_i , m is the actual message, and σ the cryptographic protection of the message.

In a one-message protocol, $\pi_{i,j}^s$ will immediately switch to `finished` state, and can no longer be activated. A responder oracle $\pi_{i,j}^t$ will be activated by a protocol message. The message will be checked according to the protocol specification. If the check succeeds, $\pi_{i,j}^t$ will switch to `accept` state.

6.1.3 Adversary

The adversary \mathcal{A} controls all the communication and is *implemented as a Turing machine, a strategy to break a DB protocol* and communicates with other processes that are described as Turing machines also, through their input and output tapes. The adversary can create many oracles of parties of its choice, and specify who wants to authenticate to whom. We model this by the adversary having access to oracles $\pi_{i,j}^s$, $s \in \mathbb{N}$. Adversary can corrupt some of the parties.

The event(s) which define a protocol break are modeled as winning events in different games, which are defined later in this section.

The capabilities of an adversary are modeled through queries: The `Send` query models that the adversary completely controls the network: *All messages are planned by \mathcal{A} who initiates the transmission of a protocol message by a process oracle of party P_i , and later plan its delivery as desired: \mathcal{A} may decide to drop the message, or store, delay or replay it, or to alter and forward it.* `Corrupt` query models real world attacks against parties, and `Tick` query models changes the clock of parties.

The adversary communicates with a process oracle via queries of the form (P_i, P_j, s, x) written on a special tape meaning that \mathcal{A} is sending message x to P_i , claiming it is from P_j in session s .

Thus messages received through a `Send` query are handled by the process oracles exactly like real protocol messages: They may be rejected, they may be answered, or they may start or terminate a protocol session. Formal description of these queries are as follows:

- $\text{Send}(\pi_{i,j}^s, m)$: The adversary can use this query to send a message m of his own choice to oracle $\pi_{i,j}^s$. The oracle will respond according to the protocol specification, depending on its internal state. If $m = \top$ consists of a special symbol \top then $\pi_{i,j}^s$ will respond with the first protocol message. For other messages the process oracle will generate the response by *running the protocol*.
- $\text{Corrupt}(P_i)$: This query reveals all keys of party P_i to \mathcal{A} . Upon receiving this query, a party P_i returns its all long lived keys shared with other parties $k_{i,j}, i, j \in \{1 \dots n\}, i \neq j$, the current state and transcript of each oracle in P_i to the adversary. That is, \mathcal{A} has access to the state of all corrupted parties and can generate all messages of this party. We assume that \mathcal{A} always has access to the state of all the helpers in the system.
- $\text{Tick}(A)$: increments the clock variable at party A , and returns it. It models the adversarial control of clocks at different parties.

6.1.4 Running the protocol in presence of the adversary

Consider a set of parties \mathbb{P} , a set of actors \mathbb{C} , with associated assigned locations in the set $\mathbb{R} \times \mathbb{R}$. A *DB (distance bounding) configuration* is specified by a tuple $C = (\mathbb{P}, \mathbb{C}, \mathbb{R} \times \mathbb{R})$. For a configuration C , running the protocol Π means performing the following experiment:

1. Choose random strings $r_G \in \{0, 1\}^*$, $r_A \in \{0, 1\}^*$ and, for each $i, j \in I$, $s \in \mathbb{N}$, a random string $r_{i,j}^s \in \{0, 1\}^*$.
2. Set $a_i = \mathcal{G}(1^\lambda, i, r_G)$ for $i \in I$, and set $a_A = \mathcal{G}(1^\lambda, A, r_G)$.
3. The adversary \mathcal{A} will run on the input $(1^\lambda, C, a_A, r_A)$. When \mathcal{A} issues a query $\text{Send}(\pi_{i,j}^s, m)$,

the process oracle $\pi_{i,j}^s$ computes a response $(m, \delta) = \Pi(1^\lambda, i, j, a, \text{loc}_{P_i}, T_{P_i}^0, r)$.

For simplicity of presentation we assume the adversary can only corrupt parties that are not running a responder oracle (verifier). The model can be extended to include cases that sessions of a party can be individually corrupted.

The adversary's oracle call must satisfy *Spatial Consistency Constraints (SCC)* that reflect physical location of participants, and the principle that information cannot travel faster than light.

Definition 6.1.2 (Spatial Consistency Constraint (SCC)) *If we have a message m written on the output tape of a party P_i located at loc_{P_i} at time t_i , then it cannot result in a correlated message m' written on the input tape of a party P_j located at loc_{P_j} at time t_j , such that -*

$$t_j - t_i < d(\text{loc}_{P_i}, \text{loc}_{P_j}) / \mathcal{C}$$

t_i, t_j are the local clock values of P_i, P_j respectively. Here correlated means $\Pr(m, m') \neq \Pr(m)\Pr(m')$.

SCC ensures that \mathcal{A} respects physical laws governing transmission of information.

Behavior of a corrupted party. We denote a corrupted party by P^* . All the LL keys, as well as current state and transcript of each oracle of a corrupted party are exposed to the adversary. However, P^* still respects the SCC and responses with correct protocol messages upon receiving a query from the adversary.

Definition 6.1.3 (δ -synchronization) *An adversary in the one shot distance bounding authentication model satisfies δ -synchronization if it never causes the clock variables of any two (uncorrupted) parties to differ by more than δ .*

In order to define the security of our one-message distance bounding protocol, we first need to define a benign adversary.

Definition 6.1.4 (Benign Adversary) *A benign adversary \mathcal{A} is an adversary that forwards messages instantaneously without modifying them.*

In the rest of this chapter, we consider an initiator oracle is always from a Prover party, and a responder oracle is always from a Verifier.

Definition 6.1.5 (One-shot Distance Bounding Protocol) *A one-shot Distance Bounding protocol authenticates a party and its location with respect to a trusted verifier. The protocol is a tuple (P, V, \mathcal{B}) , where P is a randomized prover algorithm that takes a pre-shared secret key x , V is a randomized verifier algorithm that takes pre-shared secret key y and \mathcal{B} is the distance-bound.¹ The protocol consists of one message in which the prover P initiate the protocol and the verifier V corresponds to the responder. The protocol outputs 1 indicating accept and outputs 0 indicating reject. The protocol satisfies two properties:*

- *Termination: $(\forall loc_V)$, in an execution of the protocol, the prover initiates the protocol, sends a message, and immediately switches to finished. The verifier is activated by receiving a message and finishes with an accept or reject state.*
- *p -Completeness: $(\forall loc_V, loc_P \text{ such that } d(loc_V, loc_P) \leq \mathcal{B})$, in all executions that are in presence of a benign adversary, we have $\Pr[Out_v = 1] \geq p$.*

6.2 One-Shot DB Attacks

For one message authentication protocols, the notion of matching conversation (Definition 2.3.2) in Bellare and Rogaway [BR93] was given up by Canetti and Krawczyk [CK01] and also by Schwenk (Section 3.2.2) [Sch14], since the responder is always subject to replay attack. We use Schwenk's approach in replacing the notion of matching conversation for one message authentication protocols. Schwenk had two-fold security goals for their one message authentication protocol - the responder should be protected from replay attack, and a message modified by the adversary should not be accepted by the responder. In addition to these two, our DB model has another security

¹A randomized key generation algorithm $KeyGen(1^\lambda, r)$ that takes security parameter λ and randomness r , generates secret keys x and y before the protocol execution. When symmetric key scheme is used, x is identical to y .

goal: a message originated from a far-away initiator oracle should not be accepted by the responder. An initiator oracle is far-away with respect to a responder oracle, if $d(loc_I, loc_R) > \mathcal{B}$ where loc_I and loc_R are the locations of the initiator and the responder oracles, and \mathcal{B} is the maximum allowed distance between these two parties. The last security goal has been widely studied in distance bounding literature, and is believed to be achieved by providing security against three main kind of DB attacks - Distance Fraud (Attack 3.1.1), Mafia Fraud (Attack 3.1.2) and Terrorist Fraud (Attack 3.1.3). In DF, the initiator is corrupted who attempts to shorten its distance to the responder; in MF attack, adversary \mathcal{A} intercepts messages from sending oracles, and construct their own message; in TF attack, \mathcal{A} uses a combination of corrupted initiator and helper. We give formal definition of security against these attacks later in this section.

We start by designing a security game that is used to formalize the above security goals.

Security Game G_{DB} . In this game, the challenger \mathcal{R} sets up a configuration C with n parties P_1, \dots, P_n and m helpers of these parties, and prepares ℓ protocol oracles $\pi_i, i = 1 \dots \ell$ for each party. If initiated by the adversary by a special start message, these oracles act as initiator oracles, and if initiated with a normal protocol message, they act as responder oracles. \mathcal{R} uses $KeyGen(1^\lambda)$ to generate long-lived keys (or long-lived key pairs, respectively) for each party (for each pair of parties, respectively),

\mathcal{A} may now ask up to q Send, Corrupt, and Tick queries. \mathcal{A} wins the game if - 1) there are at least two responder oracles of uncorrupted parties that accept the same message; or there is a responder oracle of an uncorrupted party that accepts a message from an uncorrupted expected sender which has not been issued by any sender oracle, or 2) \mathcal{A} succeeds in Distance Fraud, or Mafia Fraud, or Terrorist Fraud attack.

Definition 6.2.1 A DB protocol is $(C, q_s, q_c, q_t, \epsilon, \epsilon_{DF}, \epsilon_{MF}, \epsilon_{TF})$ secure, if for any configuration C , and any execution in presence of an adversary \mathcal{A} with access to q_s, q_c and q_t send, corrupt and tick queries, respectively,

1. with probability at least $1 - \epsilon$ we have that -

(a) for each responder oracle that accepts, there is exactly one uncorrupted finished initiator oracle, and

(b) for each finished uncorrupted initiator oracle, there is at most one responder oracle that accepts. And

2. the protocol is secure against DF, MF and TF attack in Definition 6.2.2, 6.2.3 and 6.2.4 respectively with probability at least $1 - \epsilon_{DF}$, $1 - \epsilon_{MF}$ and $1 - \epsilon_{TF}$ respectively.

Distance fraud (DF) attack. In this attack, a corrupted far-away initiator (*i.e.*, who is further than the distance bound \mathcal{B}) wants to convince the responder that it has a distance at least \mathcal{B} .

Security Game G_{DF} . The game is as G_{DB} with following exception- \mathcal{A} acts as a benign adversary. \mathcal{A} wins the game if, there is an uncorrupted responder oracle that accepts at least one corrupted, far-away initiator oracle.

Definition 6.2.2 (DF-security) A DB protocol Π is (τ, q, ϵ_{DF}) -secure against DF, if for any adversary \mathcal{A} that runs in time τ and asks at most q queries, the probability of winning the game G_{DF} is at most ϵ_{DF} .

Mafia Fraud (MF) attack. In the MF attack, adversary \mathcal{A} attempts to make a responder oracle accept a far-away uncorrupted initiator.

Security Game G_{MF} . The game is as G_{DB} but the winning condition is different. \mathcal{A} wins the game if, there is an uncorrupted responder oracle that accepts at least one uncorrupted, far-away initiator oracle.

Definition 6.2.3 (MF-security) A DB protocol Π is (τ, q, ϵ_{MF}) -secure against MF, if for any adversary \mathcal{A} that runs in time τ and asks at most q queries, the probability of winning the game G_{MF} is at most ϵ_{MF} .

Mafia fraud and impersonation attack. Definition 6.2.3 is general and covers Mafia fraud and impersonation attack as special cases. In Mafia fraud, there is no learning phase. The adversary

interacts with an uncorrupted initiator (through `Send` queries) and makes the responder accept. In impersonation attack the adversary uses messages of multiple uncorrupted initiators to construct a message that will be used by a helper that is located within the distance bound of \mathcal{B} to the responder to make the responder accept.

Terrorist Fraud (TF) attack. In TF attack the adversary corrupts a prover P_i^* who is outside the distance bound \mathcal{B} , and has a helper H_i who is within the distance bound, and results in the collusion of the two to succeed in the attack in the sense of resulting the verifier (responder oracle) to output `accept`. In traditional DB protocol, protection against a TF attack is by requiring that if, the attacker succeeds in TF attack, it will allow the helper to succeed in an impersonation attack by itself. The intuition is that successful TF attack must leak key information to the helper. The key point in the collusion above to be meaningful is that the computation that requires the key of the prover, and the location of the helper, are at two physically separate location that satisfy the above requirements.

Modeling this attack in our framework however is challenging. By allowing the adversary to use `Corrupt(P_i)` query, the adversary can learn the key set and state of P_i , and so effectively compute all messages that can be constructed by P_i . Assuming that the H_i algorithm is simply copying the input tape to its output and sending the associated message, success in TF attack will become straightforward: the adversary simply uses the state information of the helper and the key $k_{i,j}$ and state information of P_i^* to construct a valid protocol message from the location of the helper, and send it to the verifier using a `Send` query to the helper, with appropriate argument. In other words by allowing the adversary to obtain key information, and without any restriction on how this key information can be used by the adversary, protecting against collusion attack becomes impossible.

The main requirement of DB protocols is that the combined resources of the prover, that holds the key $k_{i,j}$, and the helper, that holds the location information cannot break the protocol *assuming the spatial separation of computations that depend on the key, and the claimed location of the prover,*

is maintained. That is, even if the adversary has the key information $k_{i,j}$, the spatial separation between the location that the key $k_{i,j}$ can be used in a computation, and the helper's location must be respected. Fortunately, our spatial consistency constraint SCC fulfill this requirement.

Security Game G_{TF} . The game is as G_{DB} but the winning condition is different. \mathcal{A} wins the game if,

1. an uncorrupted responder oracle accepts a message from an oracle $\pi_{i,j}^h$ of near-by helper H_i , while an initiator oracle $\pi_{i,j}^s$ of the corresponding prover P_i^* is far-away, and
2. SCC is preserved on the timing of: writing on output tape of P_i^* and writing on the input tape of H_i , and
3. H_i is not able to succeed in a future impersonation of prover P_i^* .

Definition 6.2.4 (TF-security) A DB protocol Π is (τ, q, ϵ_{TF}) -secure against TF, if for any adversary \mathcal{A} that runs in time τ and asks at most q queries, the probability of winning the game G_{TF} is at most ϵ_{TF} .

6.3 Beacon-based Secure One-Shot DB Scheme

In a distance bounding protocol a participant is represented by an interactive Turing Machine, that is equipped with a local clock, and has a physical location. Participant may also have secret keys: in the case of a prover or a verifier, they will have secret keys that is used for proving and verifying the distance claims.

Local clocks are all initialized at 0, but can “drift” over time. Thus a *Global Beacon (GB)* broadcasts timestamp messages to all participants that will be used by them to synchronize their local clocks.

GB broadcasts timestamp messages with frequency f per second. We assume the local clock of a participant P will tick f_p times per second and $f_p \gg f$. This means that participants expect to

receive a timestamp message every f_p/f tick of their local clock. By monitoring drift of their local clock over time, this expected arrival time of the timestamp signal can be made more accurate. Let the expected drift of the local clock tick of a participant in every f_p/f local clock ticks, be f_p^δ . That is, after synchronization of a received timestamp, the local clock ticks will increment at tick rate f_p , and *after f_p/f ticks, the drift will be in the range $(f_p/f) \pm f_p^\delta$.*

A GB timestamp message $B_t = (t, r_t)$ consists of time t that is incremented by one in consecutive timestamps, and a random string r_t that is used to ensure unforgeability of the timestamp.

When a participant receives a timestamp $B_t = (t, r_t)$, it does two things (i) it adjust its local clock to t , and (ii) appends $B_t = (t, r_t)$ to a local database. The participant will also start a timestamp verification process, which if fails the participants will declare an *attacked* state and quits.

Thus a participant effectively maintains two time values locally: (i) t_p^s that is its (GB) synchronized clock, and (ii) time t_p^ℓ that corresponds to the number of its local clock ticks after the last t_p^s . Both local times have integer values. Without adversarial interference, t_p^s has the same value for all participants and t_p^ℓ will be in a range given by $1/[(f_p/f) \pm f_p^\delta]$ seconds.

Similar to Lamport's [Lam78] model of time, messages are used to order time- in this case synchronize the participants' local clocks with the GB's clock. We also consider a *real time* that will be used to study snapshots of message transfers in the system, as well as comparing drifts of local clocks between timestamps.

In practice GB messages may take different amount of time to get to participants and so the real time of reception of timestamp is different at different participants (See Figure 6.1).

6.3.1 Global Beacon

We assume GB generates timestamp messages that satisfy the following properties:

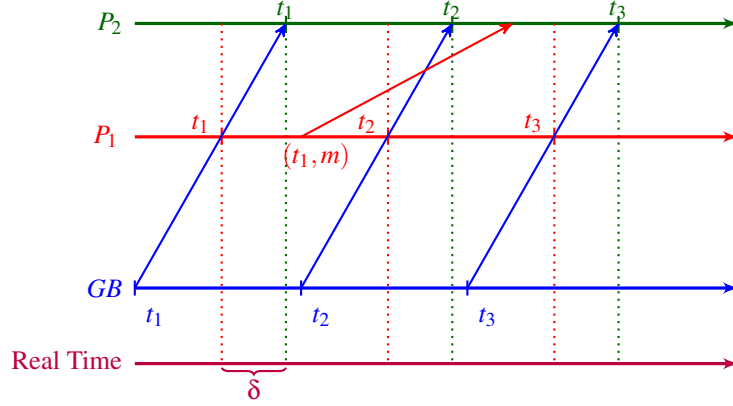


Figure 6.1: Arrival time of a timestamp at different entities. t_1, t_2, t_3 are three consecutive beacon times. GB's timestamp is received at P_2 with δ second delay compared to P_1 . P_1 receives timestamp t_1 , and forms the message (t_1, m) and sends to P_2 . Distance $(d(loc_{P_1}, loc_{P_2}))$ calculated by P_2 is $(t_2 - t_1)/\mathcal{C}$.

- **Autonomy.** The global beacon is autonomous and will run irrespective of the adversary. It generates timestamps $B_t = (t, r_t)$, consisting of a time counter t that increments by one in consecutive timestamps, and a (pseudo)-random string r_t , that is used for verification.
- **Accessibility.** The global beacon broadcasts timestamps, and the messages can be received within the region of the DB protocol.
- **Verifiability.** Timestamps are verifiable. This is because we assume the adversary controls the communication of messages and so participants must be able to verify the received timestamps.
- **Unpredictability.** We require timestamps to be unpredictable (except with negligible probability). This is achieved by using a random string r_t that is attached to the time t to be (pseudo)-random. An adversary can always guess r_t and their success chance will be 2^{-a} for an a -bit r_t . Unpredictability also implies *unforgeability*.

6.3.2 Adversary

An adversary can corrupt a participant in which case it can set the local clock of the participant to any desired value.

For non-corrupted participant, adversary can affect their local clock by manipulating timestamp messages. In particular the adversary can delay, block or modify timestamp messages.

We assume *blocking can be detected by a participant* in which case the participant will declare *attacked* and terminates the protocol. As noted earlier, assuming a local clock ticks $f_p \gg f$, a participant can calculate an expected reception time for timestamps and declare attacked state if it is not received within this time.

We assumed timestamps are verifiable and unforgeable and so tampering of timestamp will be detected by participants. With high probability the adversary's success chance of forging a new timestamp will be negligible.

The above two properties can be provided by using a broadcast authentication system that is used by GB to append a tag to each timestamp. Each participant have their verification key that allows them to verify a received timestamp. This can be constructed using shared key systems - the verification key of each participant will be different. All time signals are recorded as they arrive, but processed and verified later and offline. A protocol time is evaluated offline and when all time signals are verified.

The MAC tag that is appended to a time counter t - for example by calculating $MAC(k_{GB}, t)$ where k_{GB} is the MAC generation key of the GB- will be pseudo random and can be used as challenge.

The adversary however can delay the timestamp signal by δ_p^ℓ seconds, which correspond to the expected clock tick drift of the local clock of participant P in every f_p/f ticks (tick drift is given by the range $(f_p/f) \pm f_p^\delta$).

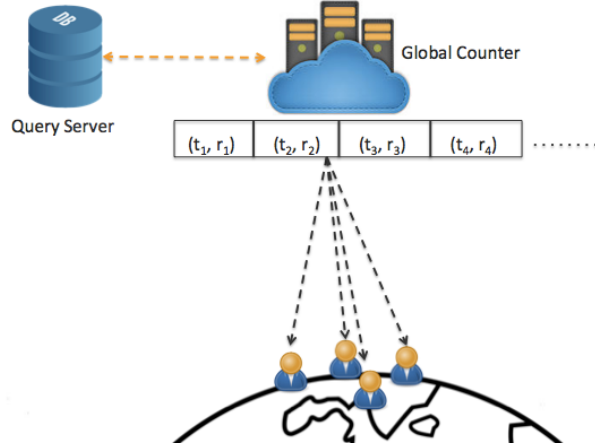


Figure 6.2: global beacon broadcasts unpredictable timestamps to users

6.4 Beacon-based One-Shot DB Construction: BShot

The basic protocol is a one-message protocol in which the prover sends a message that includes its current synchronized time, together with other data that allows the verifier to verify the identity and location claim of the prover. The verifier records the reception time of the message using its local time, and by comparing its local time with the claimed synchronized time of the prover, estimates the travel time of the message and so the distance of the prover.

To guarantee security, the prover's message will depend on their shared secret key with the verifier. We follow the approach of traditional shared key DB protocols but instead of using the secret key of the prover to generate appropriate response for the verifier's random challenges, we will take the randomness from r_t that is part of the timestamp, to construct the response. The prover's protocol message will include t which will allow the verifier to recover the same randomness and use their shared key to verify the response.

The protocol is shown in Figure 6.3.

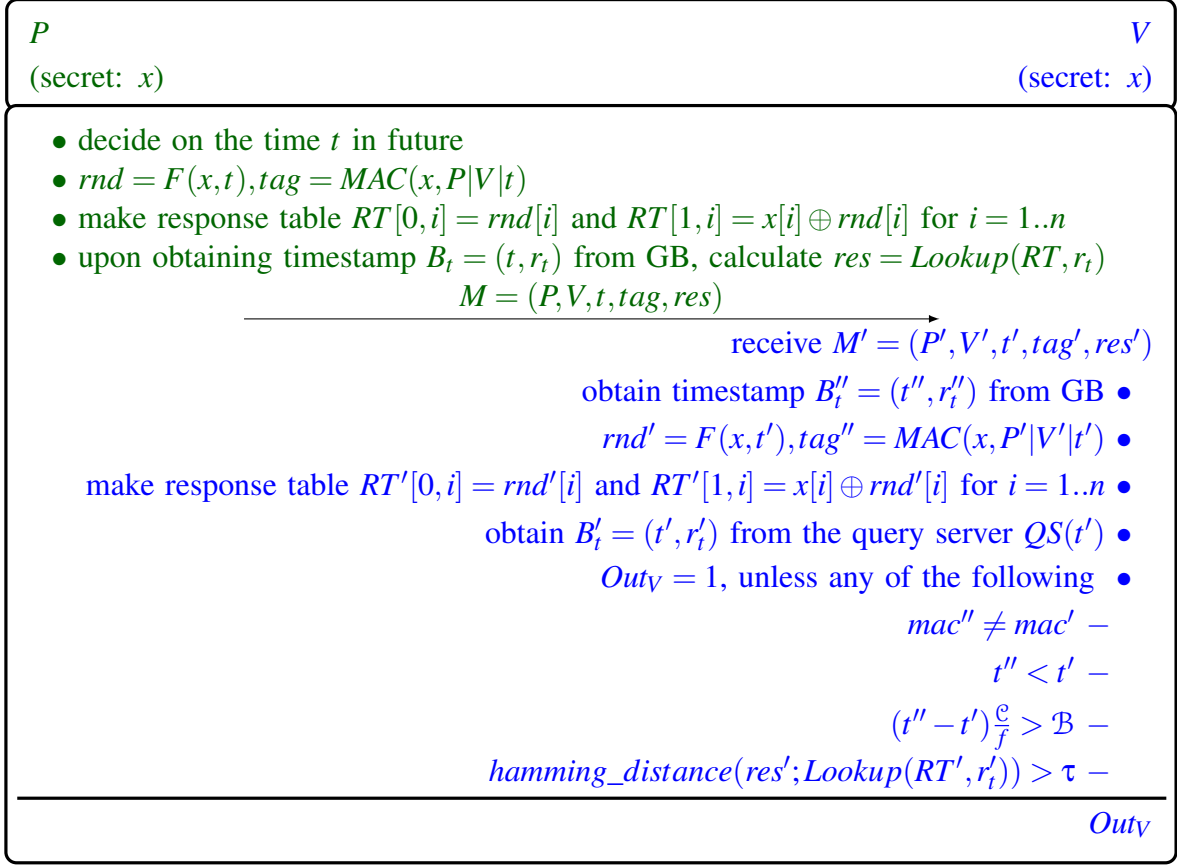


Figure 6.3: BShot: One-shot distance bounding protocol. The query server can be replaced by a local memory at each participant. c is the speed of light and f is the frequency that GB is broadcasting the timestamps.

Protocol Notations:

RT is the Response Table. It has two rows labeled by 0 and 1, and n columns: that is a $2 \times n$ binary table, $RT[i, j] \in \{0, 1\}$, and calculated as, Row 0: $RT[0, i] = rnd[i]$, and Row 1: $RT[1, i] = x[i] \oplus rnd[i]$.

$x[i], i = 1, \dots, n$ is a n -bit binary vector which is the shared secret key between P and V . $rnd[i], i = 1, \dots, n$ is an n -bit binary (pseudo) randomly generated vector calculated by $F(x, t)$ where F is a PRF (pseudo random function).

$Lookup(RT, r)$ takes an n -bit binary vector r , and outputs res , an n -bit binary vector where $res[i] =$

$RT[r[i], i]$. That is row $r[i]$ of RT is used to find the i -th response bit. $MAC(x, m)$ is an algorithm that takes a secret key x and a message m and generates a binary vector that is called tag . We use $m = P|V|t$ to show concatenation of binary strings corresponding to the identifiers of P, V , and the value of t .

Protocol Description:

P (Prover) chooses a time t in the future (when they want to send the distance claim message) and performs the pre-computation (1), that is, generating the table RT (response table).

At time t , P does the following -

- Receives beacon signal $B_t = (t, rt)$
- Uses the randomness r_t to perform computation (2)
- Generate the response res corresponding to r_t using the $Lookup()$ function
- Forms and sends the message $[P, V, t, tag, res]$ to V

The message is modified by the adversary and so V receives $[P', V', t', tag', res']$, V does the following -

- Performs steps 1 and 2 in (5) - receives beacon signal $B_t'' = (t'', rt'')$, calculates RT' using the secret key x and the received message from the prover. V also calculates tag' using the received message and the secret key x .
- Makes a query $QS(t')$ to a trusted query server that stores the timestamp messages from GB and returns the corresponding randomness r_t' for time value t' . V uses this randomness r_t' to find res' . We assume that communication to the query server is secure. Note that QS can be replaced with local memory of the verifier, that stores all received beacons in a certain time window.

- Checks according to step 4 in (5). If all true, then Step 5 (distance estimation) is performed, otherwise abort.
- If the measured distance $dist$ is within the given distance bound \mathcal{B} , accept, otherwise reject.

Note that the verification of the received timestamps can be done offline - that is while other time sensitive processes are underway.

6.4.1 BShot Accuracy

Assuming the length of r_t is at least b bits, *and assuming a benign adversary*, the above protocol can provide a secure and accurate estimate of travel time of the message (and hence distance of the prover), assuming perfectly synchronized local clocks (i.e. timestamps are received at the same real time at all participants) and negligible processing and transmission delay at the prover.

In practice the distance of GB to the provers are different and the real time of the arrival of timestamps are different. Also processing power of different provers is not ideal nor the same.

In traditional DB protocols, the accuracy of distance estimation is due to processing of response and transmission only. The prover must decode the timestamp message, calculate res and transmit the message M . Although the pre-computation of RT speeds up the processing time, decoding and transmission introduce delays that can be estimated and taken into account by the verifier. Verifier needs to consider an imperfection delay threshold for the received messages from the prover T_{max} . Practical implementations of these protocols have aimed at minimizing this delay. The accuracy of a protocol is equal to the maximum distance that the adversary can be outside the distance bound, and yet be accepted by the verifier.

In our setting, in addition to the processing time of the prover, we also need to consider different positions of participants with respect to GB and time (real time) difference in receiving the same timestamp.

Let δ^{MG} denote the maximum real time difference between arrival of message B_t at two provers in the area under consideration (See Figure 6.1). Then the protocol time threshold must allow for this difference: that is for distance upper bound \mathcal{B} and the corresponding time \mathcal{B}/\mathcal{C} , the verifier would accept responses that are received within time $\mathcal{B}/\mathcal{C} + \delta^{MG}$. Now note that if P_1 is closer to GB than P_2 by δ^{MG} , they will have $2\delta^{MG}$ advantage in making location claim with respect to P_2 .

Let δ_x^P denote the time delay due to the processing limitations of prover x , and let $\delta^{MP} = \max_{x \in \mathcal{P}} \delta_x^P$, which we consider as maximum allowed processing delay. Then the total advantage (in abusing the protocol tolerance) of a malicious prover will be $T_{max} = 2\delta^{MG} + \delta^{MP}$ seconds, and this determines the time accuracy of the location claim, which allows $(2\delta^{MG} + \delta^{MP})\mathcal{C}$ meters distance inaccuracy. The delay threshold T_{max} should be less than the time interval between two consecutive beacon messages, *i.e.*, $T_{max} \leq 1/f$. This prevents the adversary to see future beacons while the verifier is still processing past sessions.

6.4.2 BShot Practicality

Reducing protocol inaccuracy happens by reducing the value of $T_{max} = 2\delta^{MG} + \delta^{MP}$; reducing δ^{MP} happens by having fast provers in processing, and reducing δ^{MG} happens by proper positioning of GB. More specifically, let's consider a circular area with radius $r = 20m$ (food court of a shopping mall) where the beacon transmitter is installed on height $h = 50m$ (top of 10th floor), as shown in Figure 6.4.

In this setting, we have $\delta^{MG} = (D_{max} - D_{min})/\mathcal{C} = \frac{\sqrt{r^2+h^2}-h}{\mathcal{C}} \approx \frac{3.8515}{3.10^8} \approx 10^{-8}$ seconds. This geographical setting is enforcing $2\delta^{MG}\mathcal{C} = 6$ meters inaccuracy. Therefore, the more h and the less r in Figure 6.4, then we can achieve more accurate system.

As the verifier needs to consider a processing delay for the prover, it allows the faster adversary to take advantage of it. The processing time of the prover consists of (i) receiving a symbol from the global counter, (ii) computing the proper message for the verifier, and (iii) transmitting the

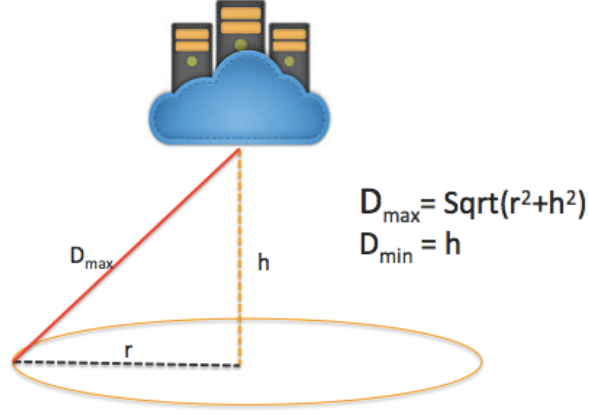


Figure 6.4: Area covered by a beacon in height.

message to the verifier.

The most efficient implementation of the prover is presented by Ranganathan *et al.* (Appendix A.2 [RTŠ⁺12]) as a hybrid digital/analog implementation, which uses an analog circuit to reply the correct response to the challenge without converting analog symbol to digital data and vice versa. In this implementation the processing time of prover is about $30ns$.

In the BShot protocol (Figure 6.3), the prover side time consuming processing elements are done prior receiving the randomness r_t . The only time-critical element is the calculation of response res , that is a table lookup operation. The whole process of receiving r_t , calculating res and transmitting res is enforcing a delay that is not reduce-able. This process is same as a normal distance-bounding protocols. The only difference is the length of the random string. However since each bit of r_t and res is designed independently from other bits, then the string of bits can be sent in parallel in order to shorten the processing delay. As a result, a careful implementation can reduce the timing of this phase to as low as $\delta^{MP} = 3 * 10^{-8}$ seconds as proposed in Ranganathan *et al.* [RTŠ⁺12].

So far, the total delay in the proposed setting is $T_{max} = (2\delta^{MG} + \delta^{MP}) = 5 * 10^{-8}$ seconds. This makes accuracy of $15 = T_{max} \cdot c$ meters. In order to achieve this accuracy, besides the computation power of the prover, we need to consider the communication factors to check feasibility. We already know that $T_{max} \leq 1/f$, so we require $f \geq 2 * 10^7$. The components of a timestamp is

$B_t = (t, r_t)$. For these components we have the following:

- **Random string r_t .** The length of random string determines the success probability of adversary in guessing a valid timestamp. The longer the random string, the less likely the adversary can succeed in forging a future timestamp. For simplicity, we use a random string of 20 bits, which gives the adversary a success probability of 2^{-20} in forging a valid timestamp.
- **Time value t .** The time value is an increasing count that is used to measure intervals. Assuming the lifetime of a timestamp is 30 minutes². Hence, we need non-duplicate counts that are enough for broadcasting for 30 minutes and the count will start over at every 30 minutes. In order to achieve $f = 2 * 10^7$, the same amount of timestamps need to be broadcasted every second, which requires $2 * 10^7$ different counts per second. For 30 minutes, $2 * 10^7 * 60 * 30 = 3.6 * 10^{10}$ indices is needed, which results in approximately 35 bits for time value field.

Hence, each timestamp will have length of $35 + 20 = 55$ bits. However, since the time value t has lots of redundancy because of being a simple counter, we can reduce the size of it in different ways, such as sending the complete string of t every 1000 beacon and for the intermediate beacons we just send the difference compared to the last full beacon. This technique reduces the size of the time value t from 40 bits to about 10 bits. As a result, the whole timestamp is 30 bits on average. Note that we consider the random string r_t as the last part of the timestamp.

Network. We need a communication channel that sends 30 bits data package from the global beacon in each $5 * 10^{-8}$ seconds. According to the above calculations, the data transfer rate is $30 * 2 * 10^7 = 580$ Mbits/s, in order to achieve the assumed security and accuracy. Current advanced wireless technology allows data to be transferred at multi Giga bit per second. For example, the IEEE 802.11ad protocol enables devices to deliver data transfer rates up to 7 Gbit/s, while

²The time model we proposed is mainly used for measuring distance of few meters to few kilometers

maintaining compatibility with existing Wi-Fi devices [NCF⁺14].

Storage. In order to replace the query server (QS) from the protocol, the verifier can store the timestamps of a time window (*e.g.*, 30 minutes) in local storage. For a 30-bit timestamp, this requires $30 \times 30 \times 60 \times 2 \times 10^7 = 135GB$ memory space with $75MB/s$ speed of writing, which is applicable with current technology.

Therefore, we can conclude this accuracy (15 meters) is applicable. Moreover, it can get even more accurate if we increase the height of global beacon, *i.e.*, h in Figure 6.4. If we put the beacon server on top of a mountain ($h = 1000$ meters), then the accuracy forced by the geography of participants reduces down to 40 cm. So the major remaining factor will be the processing delay of prover.

6.4.3 BShot Security Analysis

Lemma 31 *BShot protocol is ϵ_{DF} -resistant in Definition 6.2.2 (DF-security), where $\epsilon_{DF} = 2^{-b}$, b is length of a single row in the response table.*

Proof 31 *Consider a security game G_{DF} , where a corrupted initiator oracle $\pi_{p,v}^s$ (of party P) has been initiated by adversary \mathcal{A} and generated a message $m = (P, V, t, tag, res)$. That is, m is written on the output tape of $\pi_{p,v}^s$ at time t . Let $\pi_{p,v}^t$ be an uncorrupted receiving oracle (of party V) that reads $m' = (P', V', t', tag', res')$ at time t'' from its input tape. Now, adversary \mathcal{A} is a benign adversary in game G_{DF} , therefore $m = m'$ and m is not received by any oracle other than $\pi_{p,v}^t$. Therefore, due to the Spatial Consistency Constraint (SCC) (Definition 6.1.2), $t'' - t \geq d(loc_P, loc_V)/\mathcal{C}$. The game requires P to be far-away from V , that is $d(loc_P, loc_V) > \mathcal{B}$. Thus the only way receiving oracle $\pi_{p,v}^t$ will accept the initiator oracle $\pi_{p,v}^s$ is if the initiator oracle can forge a future timestamp t_f , $t_f > t$ in its message m .*

Recall that (protocol BShot) initiator oracle decides on the time t in future, which is input to a keyed pseudo random function (PRF) in generating the response table RT . Using a PRF restricts

the corrupted initiator from influencing the distribution of RT by selecting t maliciously. res is calculated using RT and the random string r_t (broadcast by GB), therefore the only way for a corrupted initiator oracle to forge a future timestamp in m is to guess r_t . Given the length of a single row in RT as b , $\pi_{p,v}^s$ must guess first b bit of r_t correctly for the receiving oracle $\pi_{p,v}^t$ to accept. Therefore, the winning probability ϵ_{DF} of \mathcal{A} in G_{DF} is $\epsilon_{DF} = 2^{-b}$. ■

Lemma 32 For $\epsilon_{MF} = \max(2^{-b}, \epsilon_{PRF})$, BShot protocol is ϵ_{MF} -resistant in Definition 6.2.3 (MF-security).

Proof 32 In a security game G_{MF} , let $\pi_{p,v}^s$ (of party P) be a uncorrupted initiator oracle far away from a receiving oracle $\pi_{p,v}^t$ of party V . Let $\pi_{p,v}^a$ be an oracle of close-by actor A . In MF attacks, a common strategy of the adversary \mathcal{A} is to act as a verifier for the far-away prover, obtain responses and then use the gathered information to make the verifier accept a close-by actor. In BShot, \mathcal{A} could try to determine the response table of the uncorrupted oracle $\pi_{p,v}^s$ by sending protocol initiating messages to the oracle. However, a keyed PRF is used in generating the response table which takes a future time t as input, which restricts the adversary from determining the response table. Therefore, security of BShot is reduced to security of the PRF.

A second strategy for \mathcal{A} would be to utilize the close-by actor's location to replace the old timestamp with a new one in the message originated in the far-away initiator oracle. That is, if $\pi_{p,v}^s$ generates a message $m = (P, V, t, tag, res)$ intended for $\pi_{p,v}^t$, the actor oracle $\pi_{p,v}^a$ replaces (t, res) with newer values (t', res') . However, in order to compute res' correctly, the response table RT is required $res' = \text{lookup}(RT, r'_t)$. Now, res' can either be guessed by the adversary with probability 2^{-b} , or they can try to determine the response table. In the latter case, security of BShot is reduced to security of the PRF that is used to generate RT . Therefore, we have $\epsilon_{MF} = \max(2^{-b}, \epsilon_{PRF})$. ■

Lemma 33 BShot protocol is ϵ_{TF} -resistant in Definition 6.2.4 (TF-security), where $\epsilon_{TF} = 2^{-b}$.

Proof 33 In a security game G_{TF} , let $\pi_{p,v}^s$ (of party P) be a corrupted initiator oracle far away from a receiving oracle $\pi_{p,v}^t$ of party V . Let $\pi_{p,v}^h$ be an oracle of close-by helper H of party P .

Now, the $\text{Corrupt}(P)$ query enables the adversary \mathcal{A} to learn the key set and state of P , and so efficiently compute all messages that can be constructed by P . Let us consider following strategy of the adversary: H simply copies its input tape to its output and sends associated message, and adversary uses the state information of H and the key x of the initiator oracle to construct a valid protocol message from the location of the helper, and send it to the verifier using a $\text{Send}()$ query to the helper, with appropriate argument. However, this straightforward attack is restricted by the Spatial Consistency Constraint (SCC, Definition 6.1.2). The second winning condition in the game G_{TF} is that SCC should be preserved between writing on output tape of $\pi_{p,v}^s$ and writing on the input tape of $\pi_{p,v}^h$. That is, even if the adversary has the key information x , the spatial relation between the location of the key and the helper must be respected. This effectively reduce this attack to a distance fraud attack, which has success probability 2^{-b} .

A second strategy by \mathcal{A} would be: in BShot , the corrupted far-away oracle $\pi_{p,v}^s$ chooses a time t in a far-enough future (such that $\pi_{p,v}^s$ will be treated as close-by), generates the response table for this t and hand over the response table to the close-by helper oracle $\pi_{p,v}^h$. The helper oracle is now responsible to obtain the timestamp $B_t = (t, r_t)$ and generate and send message m to the responder oracle. However, doing this enables the helper oracle correctly compute the secret key x , by bitwise XORing the two rows of the response table. This violates the third winning condition of G_{TF} , which is - the helper must not be able to succeed in a future impersonation of the prover (initiator). Therefore, the helper must not be provided with the response table, and can only guess res with probability 2^{-b} . Thus, we have $\epsilon_{TF} = 2^{-b}$. ■

Theorem 5 BShot is a $(C, q_s, q_c, q_t, \epsilon, \epsilon_{DF}, \epsilon_{MF}, \epsilon_{TF})$ secure DB protocol in Definition 6.2.1, where $\epsilon \leq n^2 \cdot \max(2^{-b}, \epsilon_{PRF})$, $\epsilon_{DF} = 2^{-b}$, $\epsilon_{MF} = \max(2^{-b}, \epsilon_{PRF})$ and $\epsilon_{TF} = 2^{-b}$.

Proof 34 Consider the security game G_{DB} . For the moment, let us ignore the winning condition that involves distance bounding attacks (i.e., DF, MF and TF). Let G_0 be the original game, where adversary \mathcal{A} attempts to make an oracle accept a message m which is either faked, or has already been accepted by a different oracle. Thus we have $\epsilon_0 = \epsilon$.

Let P be the initiator party and V be the responder party in game G_1 . P, V share secret x . We make a guess that \mathcal{A} will be successful in making one responder oracle $\pi_{p,v}^t$ accept a message m which is either faked, or has already been accepted by a different oracle. If our guess is wrong, we abort the game, and the adversary loses. Since there are n parties in the system, \mathcal{A} 's winning probability is reduced by a factor n^2 . Thus we have, $n^2 * \epsilon_1 = \epsilon_0$.

In G_2 , we abort the game if adversary \mathcal{A} can forge a valid response res for a time value t in message $m = (P, V, t, tag, res)$ for the key x . This may happen with probability $\max(2^{-b}, \epsilon_{PRF})$ (i.e., correctly guessing probability is 2^{-b}). The simulator replaces all PRF computations involving key x with calls to a PRF challenger \mathcal{R}_{PRF} which uses a randomly chosen PRF key x ; if \mathcal{A} forges a valid response res for a time value t which has not been queried from the PRF challenger, then we have broken the PRF challenge. Thus we have $\epsilon_1 \leq \epsilon_2 + \max(2^{-b}, \epsilon_{PRF})$.

Now that we have excluded PRF forgeries in this game, we are left with one-way messages $m = (P, V, t, tag, res)$ which were generated by non-corrupted oracles, where only the value t may be influenced by adversary. Since this influence is detected by the verifier (because of MAC verification), condition 1-(a) of Definition 6.2.1 is always true (i.e., holds with probability 1), and we are left with condition 1-(b).

If \mathcal{A} tries to send m to any oracle of a party $V' \neq V$, V' will not accept because the target identity is different (i.e., MAC verification will fail). In our DB model, we do not consider the receiver and initiator oracle being in the same party. Thus, also condition 1-(b) is fulfilled (i.e., holds with probability 1). We have $\epsilon_2 = 0$, therefore $\epsilon \leq n^2 \max(2^{-b}, \epsilon_{PRF})$.

$\epsilon_{DF}, \epsilon_{MF}$ and ϵ_{TF} are determined in Lemma 31, 32 and 33 respectively. ■

6.5 Concluding Remarks

We proposed the first one message distance-bounding model (called One-Shot DB), and construction (called BShot protocol) with formal definitions and security proof. With this approach we solved some of the major problems in applying distance-bounding in real life:

- Despite traditional DB protocols, participants of One-Shot DB can run the protocol even while they are moving.
- The DB system becomes highly scalable in a populated area for two reasons; first reducing the number of fast phase messages from 2ℓ to just one message, and second making a single beacon to be usable by all users in the area, instead of the challenge message in traditional DB protocols that are generated for a single user.
- Verifier can check the location of provers continuously rather than one time checking in traditional DB protocols.

Implementation of a prover is still a hard problem in this literature. Although our construction does not improve the solutions of this problem compared to traditional DB, our model opens the horizon to consider other techniques to solve this problem. For example, if we can replace randomness of the beacon with randomness of the environment noise, then we can remove the prover processing delay, which is the major limitation for achieving high accuracy. This is based on the assumption that two parties are detecting similar environmental noise as long as they are close to each other. We left this as a future research.

Chapter 7

Concluding Remarks

In this thesis we looked at three main problems: Public-Key distance-bounding, Anonymous distance-bounding and One-Shot distance-bounding.

In public-key DB, we proposed a new formal model (DBID) for distance-bounding protocols, inline with the cryptographic identification schemes. For the first time, we allow the communication channel to mix multiple messages into one message at the receiver, if they are received at the same time. This is more realistic wireless environment and makes stronger adversary. Moreover, we allow the adversary to have access to directional antenna. We show that this additional capability allows to break security of protocols that had been proven secure.

We propose a public-key DB construction (POXY) that is secure in this new model. Moreover, we prove that the existing public-key DB construction, ProProx, is in fact secure in this model. These two constructions use different cryptosystems.

We consider optimizing the public-key protocols -specially reducing the redundancy of the fast phase- as future work.

In anonymous DB, we show the security challenges that arise when identity information is not directly used in DB protocol. We proposed a new model that captures all known attacks and a construction with provable security in this model. We show that using directional antenna, the adversary can break all existing anonymous DB protocols. Moreover, we propose a new attack (called collusion attack), where the provers of multiple users collude to deceive the verifier.

We proposed two constructions for different cryptosystems that convert public-key DB protocols to anonymous DB protocols. These constructions are modular and can use similar components

that follows the designed cryptosystem. These two protocols are the first that are resistant against all distance-bounding attacks, including directional antenna attacks.

One of the presented AnonDB protocols (*i.e.*, dbid2an^P) does not support transcript opening and user revocation. Designing an AnonDB scheme that uses Pedersen commitment scheme and facilitates these methods is considered as future work.

In One-Shot DB, we proposed the first one message distance-bounding model (called One – Shot DB), and propose a construction (called BShot protocol). With this approach we solved some of the major problems in applying distance-bounding in real life: (i) participants can run the protocol even when they are moving, (ii) the DB system becomes highly scalable in a populated area because of reducing the number of fast phase messages from 2ℓ to just one message, and also making a single beacon to be usable by all users in the area, instead of the challenge message in traditional DB protocols that are generated for a single user. (iii) the verifier can check the location of provers continuously rather than one time checking in traditional DB protocols, since the cost of checking is dramatically reduced.

One-Shot DB is a new domain and we can consider numerous future works. Here we mention three of them; (1) Our construction still suffers from the need for implementation of a prover that can reply instantly. Designing a scheme that removes this limitation is still an open problem. (2) In our construction, we need a trusted third party for generating the random beacons. Designing a system that derives the random fast phase challenges from the environment (*e.g.*, using environment wireless noise) is an open problem. (3) In our model, we are considering time synchronicity, which is hard and complicated to model. However in distance-bounding, we only need a form of location synchronicity. If we can use a technology that allows us to gain location synchronicity of two entities, such as wireless channel state information (CSI), then we might have simpler model and construction without any third party.

Bibliography

- [ABG⁺17] Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gérard, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 800–814. ACM, 2017.
- [ABK⁺11] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 2011.
- [ALM11] Gildas Avoine, Cédric Lauradoux, and Benjamin Martin. How secret-sharing can defeat terrorist fraud. In *Proceedings of the fourth ACM conference on Wireless network security*, pages 145–156. ACM, 2011.
- [ARS16] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2016.
- [ASN14] Ahmad Ahmadi and Reyhaneh Safavi-Naini. Privacy-preserving distance-bounding proof-of-knowledge. In *16th ICICS*, 2014.
- [ASN17a] Ahmad Ahmadi and Reihaneh Safavi-Naini. Distance-bounding identification. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 202–212. INSTICC, SciTePress, 2017.
- [ASN17b] Ahmad Ahmadi and Reyhaneh Safavi-Naini. Directional distance-bounding identification. In *Information Systems Security and Privacy*, volume 867 of *Communications in Computer and Information Science*. Springer International Publishing,

2017.

- [ASNRA18] Ahmad Ahmadi, Reyhaneh Safavi-Naini, and Mamunur Rashid Akand. New attacks and secure design for anonymous distance-bounding. In *Australasian Conference on Information Security and Privacy*. Springer, 2018.
- [AT09] Gildas Avoine and Aslan Tchamkerten. An efficient distance bounding RFID authentication protocol: balancing false-acceptance rate and memory requirement. In *Information Security*, pages 250–261. Springer, 2009.
- [BB04] Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge protocols to avoid terrorist fraud attacks. Technical report, Technical report, Institut Eurecom, France, 2004.
- [BBM⁺13] Aslı Bay, Ioana Boureanu, Aikaterini Mitrokotsa, Iosif Spulber, and Serge Vaude-
nay. The bussard-bagga and other distance-bounding protocols under attacks. In *Information Security and Cryptology*, pages 371–391. Springer, 2013.
- [BC94] Stefan Brands and David Chaum. Distance-bounding protocols. In *Advances in Cryptology–EUROCRYPT’93*, pages 344–359. Springer, 1994.
- [BCEP04] Emmanuel Bresson, Olivier Chevassut, Abdelilah Essiari, and David Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. *Computer Communications*, 27(17):1730–1737, 2004.
- [BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428. ACM, 1998.
- [BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryp-*

tology?CRYPTO'98, pages 26–45. Springer, 1998.

- [BF09] Manuel Barbosa and Pooya Farshim. Security analysis of standard authentication and key agreement protocols utilising timestamps. In *International Conference on Cryptology in Africa*, pages 235–253. Springer, 2009.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988.
- [BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *Advances in Cryptology-CRYPTO'89*, 1990.
- [BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology – CRYPTO'92*, pages 390–420. Springer, 1992.
- [BGG⁺16] Xavier Bultel, Sébastien Gambs, David Gérard, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *WiSec '16*, 2016.
- [Bir40] Garrett Birkhoff. *Lattice theory*, volume 25. American Mathematical Soc., 1940.
- [BMK10] Jack Burbank, David Mills, and William Kasch. Network time protocol version 4: Protocol and algorithms specification. *Network*, 2010.
- [BMV12] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. On the pseudorandom function assumption in (secure) distance-bounding protocols. In *Progress in Cryptology-LATINCRYPT 2012*, pages 100–120. Springer, 2012.
- [BMV13a] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Practical & provably secure distance-bounding. In *The 16th Information Security Conference*, 2013.
- [BMV13b] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Secure & lightweight

distance-bounding. In *International Workshop on Lightweight Cryptography for Security and Privacy*, 2013.

- [BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 614–629. Springer, 2003.
- [BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Annual international cryptology conference*, pages 232–249. Springer, 1993.
- [Buc04] Johannes Buchmann. *Introduction to cryptography*. Springer Science & Business Media, 2004.
- [ČBH03] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Sector: secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 21–32. ACM, 2003.
- [CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In *Theory of Cryptography*, pages 380–403. Springer, 2006.
- [Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.
- [CHKM06] Jolyon Clulow, Gerhard P Hancke, Markus G Kuhn, and Tyler Moore. So near and yet so far: Distance-bounding attacks in wireless networks. In *Security and Privacy in Ad-Hoc and Sensor Networks*, pages 83–97. Springer, 2006.
- [CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and*

Applications of Cryptographic Techniques, pages 453–474. Springer, 2001.

- [CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO’04*, pages 56–72, 2004.
- [CRSC12] Cas Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Capkun. Distance hijacking attacks on distance bounding protocols. In *Security and Privacy*, 2012.
- [Dam02] Ivan Damgård. On Σ -protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2002.
- [DDP06] Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. In *Advances in Cryptology-EUROCRYPT 2006*, pages 555–572. Springer, 2006.
- [Des88] Yvo Desmedt. Major security problems with the unforgeable(forge-)fiat-shamir proofs of identity and how to overcome them. In *Securicom’88*, 1988.
- [DFKO11] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding rfid protocols. In *International Conference on Information Security*, pages 47–62. Springer, 2011.
- [DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 1976.
- [DLYZ11] Robert H Deng, Yingjiu Li, Moti Yung, and Yunlei Zhao. A zero-knowledge based framework for rfid privacy. *Journal of Computer Security*, 19(6):1109–1146, 2011.
- [FDC11] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.
- [FO13a] Marc Fischlin and Cristina Onete. Subtle kinks in distance-bounding: an analysis of

- prominent protocols. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 195–206. ACM, 2013.
- [FO13b] Marc Fischlin and Cristina Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *Applied Cryptography and Network Security*, pages 414–431, 2013.
- [For87] Lance Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 204–209. ACM, 1987.
- [GAA11] Ali Özhan Gürel, Atakan Arslan, and Mete Akgün. Non-uniform stepping approach to RFID distance bounding problem. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 64–78. Springer, 2011.
- [Gen04] Rosario Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *Annual International Cryptology Conference*, 2004.
- [GKL⁺14] Sébastien Gambs, Marc-Olivier Killijian, Cédric Lauradoux, Cristina Onete, Matthieu Roy, and Moussa Traoré. Vssdb: A verifiable secret-sharing and distance-bounding protocol. In *International Conference on Cryptography and Information security*, 2014.
- [GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, pages 281–308, 1988.
- [GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity

- of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
- [Gol01] Oded Goldreich. *Foundations of cryptography: volume 1, basic techniques*. Cambridge university press, 2001.
- [Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
- [GOR14] Sébastien Gambs, Cristina Onete, and Jean-Marc Robert. Prover anonymous and deniable distance-bounding authentication. In *ASIA CCS '14*, 2014.
- [GQ88] Louis C Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *EUROCRYPT '88*, 1988.
- [Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *International Workshop on Public Key Cryptography*, pages 145–160. Springer, 2003.
- [Han11] Gerhard P Hancke. Design of a secure distance-bounding channel for rfid. *Journal of Network and Computer Applications*, 34(3):877–887, 2011.
- [Han12] G.P. Hancke. Distance-bounding for RFID: Effectiveness of 'terrorist fraud' in the presence of bit errors. In *RFID-Technologies and Applications (RFID-TA), 2012 IEEE International Conference on*, pages 91–96, 2012.
- [HK05] Gerhard P Hancke and Markus G Kuhn. An RFID distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 67–73. IEEE, 2005.
- [Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.
- [HPO13] Jens Hermans, Roel Peeters, and Cristina Onete. Efficient, secure, private distance bounding without key updates. In *Proceedings of the sixth ACM conference on Se-*

- curity and privacy in wireless and mobile networks*, pages 207–218. ACM, 2013.
- [HPVP11] Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A new RFID privacy model. In *Computer Security–ESORICS 2011*, pages 568–587. Springer, 2011.
- [JP02] Markus Jakobsson and David Pointcheval. Mutual authentication for low-power mobile devices. In *Financial Cryptography*, pages 178–195. Springer, 2002.
- [JW09] Ari Juels and Stephen A Weis. Defining strong privacy for rfid. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):7, 2009.
- [KA09] Chong Hee Kim and Gildas Avoine. RFID distance bounding protocol with mixed challenges to prevent relay attacks. In *Cryptology and Network Security*, pages 119–133. Springer, 2009.
- [KAK⁺08] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife rfid distance bounding protocol. In *International Conference on Information Security and Cryptology*, pages 98–115. Springer, 2008.
- [KH06] Kaoru Kurosawa and Swee-Huay Heng. The power of identification schemes. In *Public Key Cryptography-PKC 2006*, pages 364–377. Springer, 2006.
- [KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.
- [Kru09] John Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, pages 391–399, 2009.
- [Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.
- [LM13] Vadim Lyubashevsky and Daniel Masny. Man-in-the-middle secure authentication schemes from lpn and weak prfs. In *Advances in Cryptology – CRYPTO’13*, pages

308–325. Springer, 2013.

- [MLDL09] Changshe Ma, Yingjiu Li, Robert H Deng, and Tieyan Li. Rfid privacy: relation between two notions, minimal condition, and efficient construction. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 54–65. ACM, 2009.
- [MP08] Jorge Munilla and Alberto Peinado. Security analysis of tu and piramuthu’s protocol. In *New Technologies, Mobility and Security, 2008. NTMS’08.*, pages 1–5. IEEE, 2008.
- [MPLDV13] Aikaterini Mitrokotsa, Pedro Peris-Lopez, Christos Dimitrakakis, and Serge Vaude-
nay. On selecting the nonce length in distance-bounding protocols. *The Computer Journal*, 2013.
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.
- [NCF⁺14] Thomas Nitsche, Carlos Cordeiro, Adriana B Flores, Edward W Knightly, Eldad Perahia, and Joerg C Widmer. Ieee 802.11 ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi. *IEEE Communications Magazine*, 52(12):132–141, 2014.
- [NFHF09] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *International Workshop on Public Key Cryptography*, pages 463–480. Springer, 2009.
- [NSMSN08] Ching Yu Ng, Willy Susilo, Yi Mu, and Rei Safavi-Naini. Rfid privacy models revisited. In *Computer Security-ESORICS 2008*, pages 251–266. Springer, 2008.
- [NV08] Ventzislav Nikov and Marc Vauclair. Yet another secure distance-bounding protocol. *IACR Cryptology ePrint Archive*, 2008:319, 2008.

- [NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437. ACM, 1990.
- [Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO’91*, pages 129–140. Springer, 1992.
- [PH12] Roel Peeters and Jens Hermans. Wide strong private RFID identification based on zero-knowledge. *IACR Cryptology ePrint Archive*, 2012:389, 2012.
- [PHS03] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of computer security*. Springer, 2003.
- [PV08] Radu-Ioan Paise and Serge Vaudenay. Mutual authentication in RFID: security and privacy. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 292–299, 2008.
- [RČ08] Kasper Bonne Rasmussen and Srdjan Čapkun. Location privacy of distance bounding protocols. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 149–160. ACM, 2008.
- [RC10] Kasper Bonne Rasmussen and Srdjan Capkun. Realization of rf distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010.
- [RNTS07] Jason Reid, Juan M Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 204–213. ACM, 2007.
- [Ros98] Christian Eric Ross. Vehicle passive keyless entry and passive engine starting system, May 12 1998. US Patent 5,751,073.
- [RTŠ⁺12] Aanjan Ranganathan, Nils Ole Tippenhauer, Boris Škorić, Dave Singelée, and Srd-

- jan Čapkun. Design and implementation of a terrorist fraud resilient distance bounding system. In *Computer Security–ESORICS 2012*, pages 415–432. Springer, 2012.
- [Sch91] C P Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 1991.
- [Sch14] Jörg Schwenk. Modelling time for authenticated key exchange protocols. In *European Symposium on Research in Computer Security*, pages 277–294. Springer, 2014.
- [SP07a] Dave Singelee and Bart Preneel. Distance bounding in noisy environments. In *Security and Privacy in Ad-hoc and Sensor Networks*, pages 101–115. Springer, 2007.
- [SP07b] Dave Singelee and Bart Preneel. Key establishment using secure distance bounding protocols. In *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–6. IEEE, 2007.
- [STLBH11] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J-P Hubaux. Quantifying location privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 247–262, 2011.
- [Tip12] Nils Ole Tippenhauer. *Physical-Layer Security Aspects of Wireless Localization*. PhD thesis, ETH, 2012.
- [TP07] Yu-Ju Tu and Selwyn Piramuthu. RFID distance bounding protocols. In *First International EURASIP Workshop on RFID Technology, Vienna, Austria (September 2007)*, 2007.
- [Vau07] Serge Vaudenay. On privacy models for RFID. In *Advances in Cryptology–Asiacrypt 2007*, pages 68–87. Springer, 2007.
- [Vau13] Serge Vaudenay. On modeling terrorist frauds. In *Provable Security*. Springer, 2013.
- [Vau14] Serge Vaudenay. Proof of proximity of knowledge. *IACR Eprint*, 695, 2014.

- [Vau15] Serge Vaudenay. Private and secure public-key distance bounding. In *Financial Cryptography and Data Security*, pages 207–216. Springer, 2015.
- [Vau16] Serge Vaudenay. Privacy failure in the public-key distance-bounding protocols. *IET Information Security*, 10(4):188–193, 2016.
- [VBM⁺13] Serge Vaudenay, Ioana Boureanu, Aikaterini Mitrokotsa, et al. Practical & provably secure distance-bounding. In *The 16th Information Security Conference*, 2013.
- [WC01] Duncan S Wong and Agnes H Chan. Mutual authentication and key exchange for low power wireless communications. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 1, pages 39–43. IEEE, 2001.
- [Yao82] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

Appendix A

Extra Literature Review

In this chapter we discuss the works that are indirectly related to the problem statements of this thesis. The topics of this chapter have either less priority compared to Chapter 3, or are used as application in the body of thesis.

In Section A.1 we review the symmetric key distance bounding literature. In Section A.2 we review the difficulties implementation of a distance-bounding protocol on physical circuit. And In Section A.3 we discuss the privacy concepts in distance-bounding literature.

A.1 Symmetric Distance-Bounding

Hancke-Kuhn [HK05]

Hancke-Kuhn proposed the classic DB model, which is used in most of DB protocols. In this model, each pair of prover-verifier is given a shared-key and the prover is trying to convince the verifier that she knows the shared-key, while she is close to the verifier. Verifier sends fresh challenge bits, and the prover replies with the proper response as fast as possible in a *fast-exchange* phase.

Figure A.1 shows the scheme, which considers noisy channels for the first time in the literature. This protocol is designed to be secure against *DF* and *MF* attacks.

Reid *et al.* [RNTS07]

Reid *et al.* extended Hancke-Kuhn's protocol (Figure A.2) to be secure against *TF* attack, in which the prover is far away from the verifier, but there is a close adversary who is collaborating with the prover to succeed in the protocol. In a *TF* scenario, the prover sends the response table (i.e.

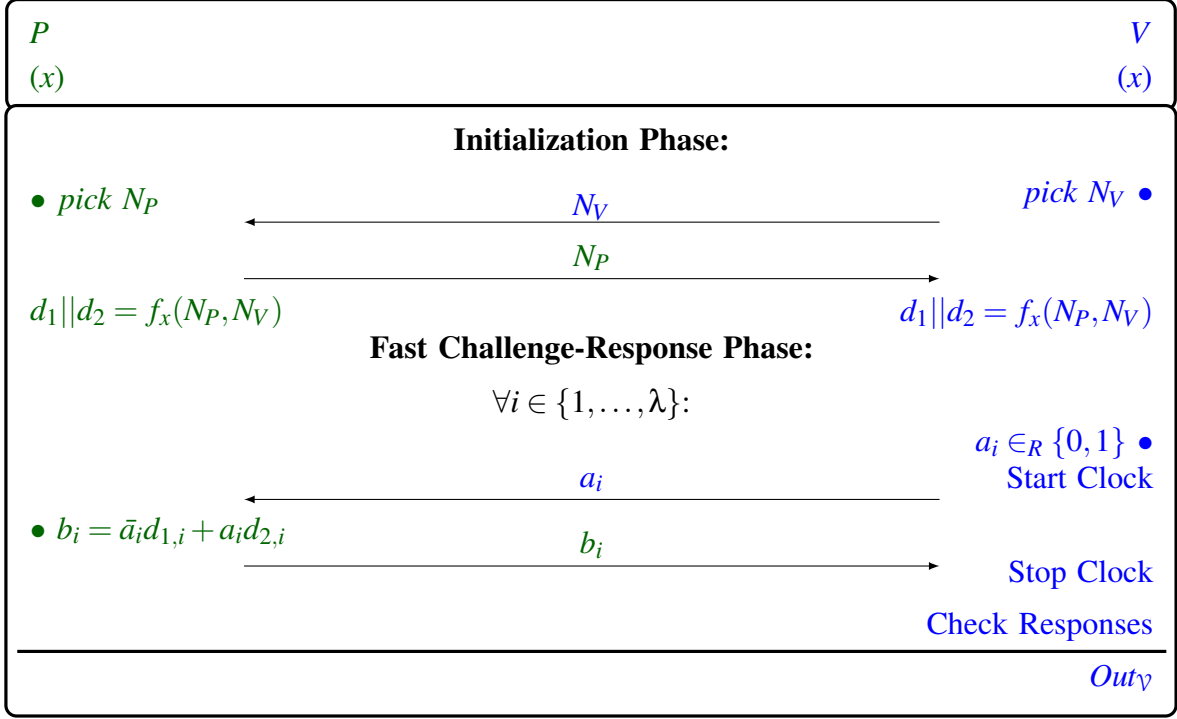


Figure A.1: Hancke-Kuhn Protocol [HK05]. $f_x()$ is a pseudo random function.

proper answer for each challenge) to the adversary prior the *fast-exchange* phase. And after that, the adversary can answer to challenges of verifier successfully.

Based on this protocol, a malicious prover who is helping the adversary with the response table, will end up leaking her own secret-key to the helper. Unfortunately Bay *et al.* [BBM⁺13] showed that this protocol becomes vulnerable to a MiM attack [KAK⁺08], which makes the protocol to be vulnerable against *MF* attack.

The idea of the attack as shown by Bay *et al.* [BBM⁺13] is that the adversary relays the communication between a close prover and a verifier, but flips one challenge a_i . The value b_i which is sent as a response to the verifier is selected at random. Therefore, the adversary learns the response of a_i from prover, and by seeing the final output of the verifier (acceptance or rejection), the adversary deduces what is the correct answer to $1 - a_i$. So, he learns the i^{th} bit of d_1 and d_2 , which deduces x_i . He can repeat this for each i and infer x . Then, the attack phase just impersonates the prover. Other instances of this protocol, where $d_2 = d_1 \oplus x$ get replaced by addition modulo some prime

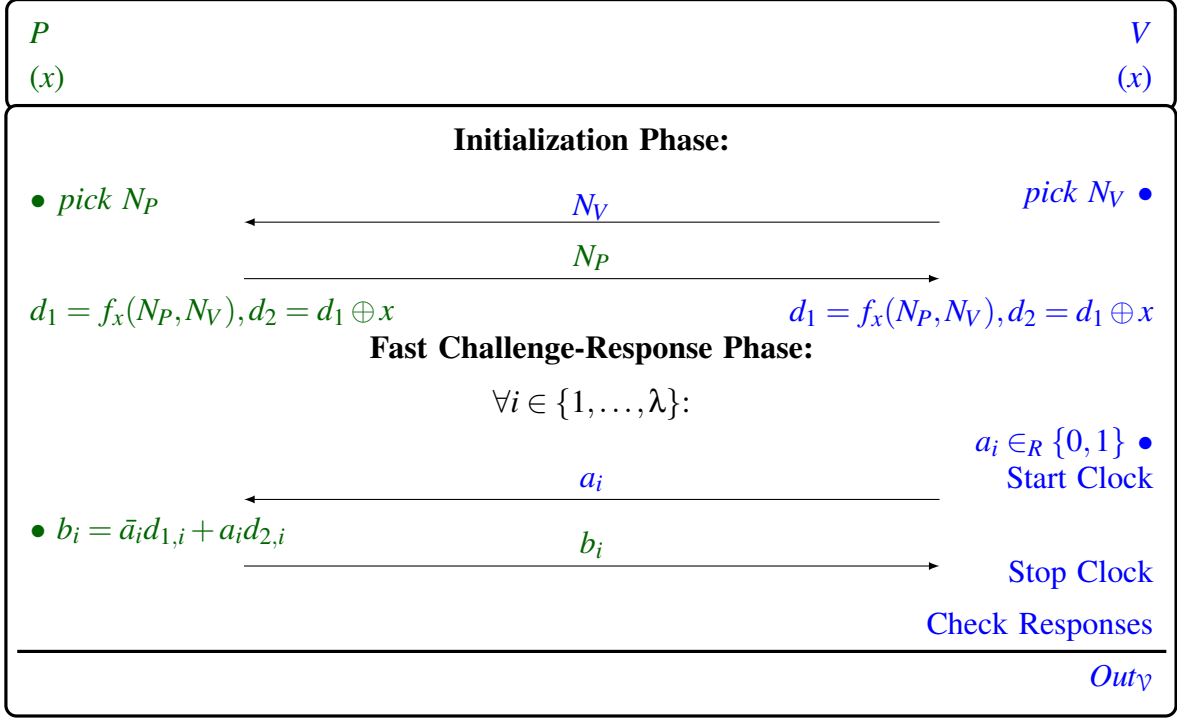


Figure A.2: Reid *et al.* distance-bounding protocol [RNTS07]. $f_x()$ is a pseudo random function.

q or addition with a random factor, can also be broken, as shown in [BBM⁺13]. However, it is suggested by Mitrokovtsa *et al.* [MPLDV13] to step back to Hancke-Kuhn's idea in calculating d_1 and d_2 (i.e. $d_1 || d_2 = f_x(N_P, N_V)$), but the prover releases $(R, x \oplus h_R(d_1, d_2))$ as an extra information, where R is a random value and $h()$ is a universal hash function. Unfortunately, the security proof of this idea is not provided.

Avoine *et al.* [ALM11]

Avoine *et al.* proposed the TDB protocol (Figure A.3) that addresses the problem of MiM attack that was brought up by Kim *et al.* [KAK⁺08]. TDB protocol [ALM11] addresses this problem by adding an additional state to the challenge (i.e. $a_i \in \{0, 1, 2\}$). The idea is to do a threshold secret-sharing for splitting x_i into three pieces, such that avoids leakage by having just two shares.

The main assumption of TDB protocol is that the function f is a pseudo-random function, but Boureau *et al.* [BMV12] showed that this assumption is not enough and they presented a PRF,

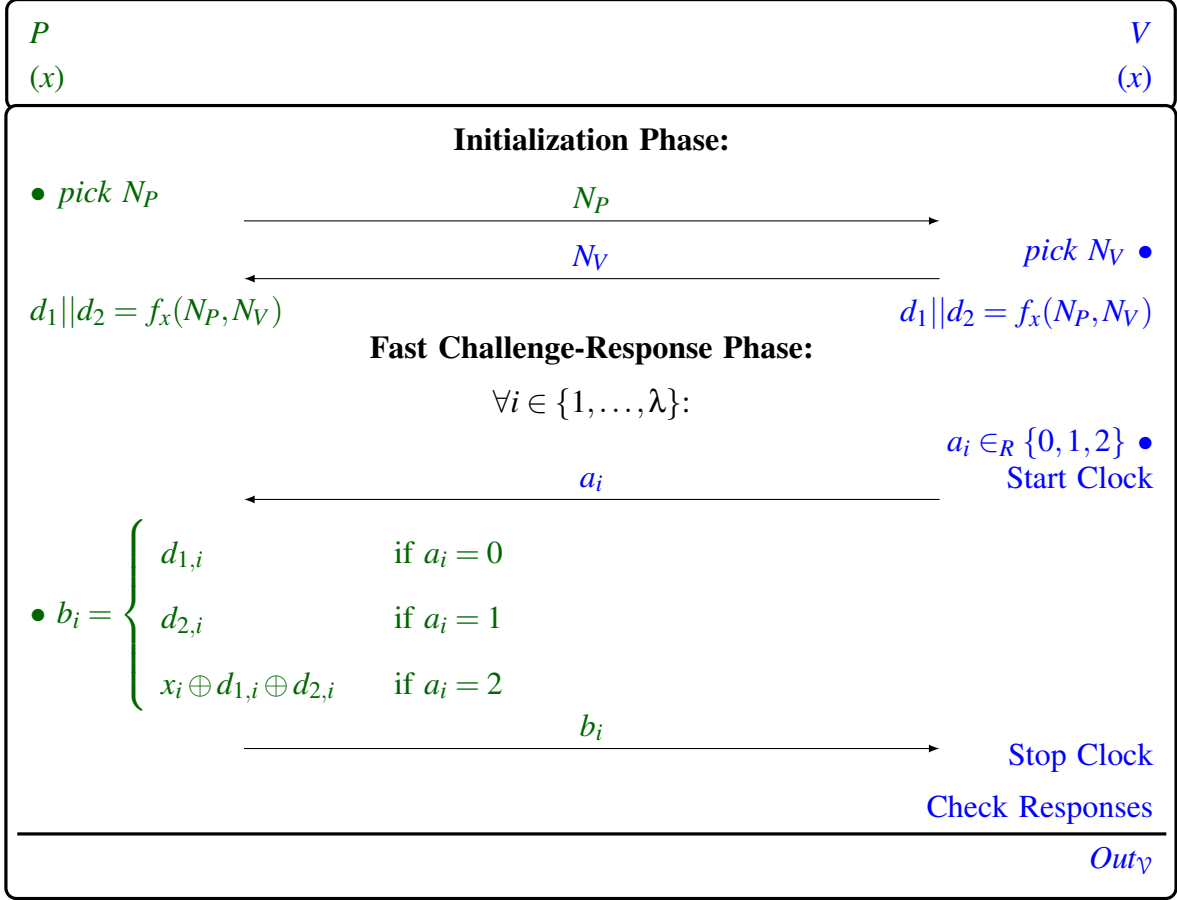


Figure A.3: TDB Protocol [ALM11]

which makes the protocol insecure. The following PRF f is constructed, given a PRF g ;

$$f_x(N_p, N_v) = \begin{cases} x || x & \text{if } N_p = x \\ g_x(N_p, N_v) & \text{otherwise} \end{cases}$$

By having this PRF f , a malicious prover can choose $N_p = x$ and make all the responses to be the same, which allows the prover to be able to send the response before getting the challenge from verifier. The protocols that can be broken by this attack is shown in Table A.1.

Boueanu *et al.* [BMV12] solved this problem by using PRF Masking, as shown in Figure A.4.

In order to handle the noise in the communication channel, the idea in many protocols is to tolerate some limited amount of wrong prover responses. Hancke [Han12] showed, this idea makes this protocol vulnerable to *TF* attack. In this attack, the malicious prover runs the initial phase, and

Protocol	Vulnerable to DF	Vulnerable to MiM
Hancke-Kuhn [HK05]	×	-
Reid <i>et al.</i> [RNTS07]	×	×
Avoine-Tchamkerten [AT09]	×	×
Swiss-Knife [KAK ⁺ 08]	-	×
Dürholz <i>et al.</i> [DFKO11]	×	-
TDB [ALM11]	×	-

Table A.1: Vulnerable DB protocols against PRF Programming Techniques [BMV12].

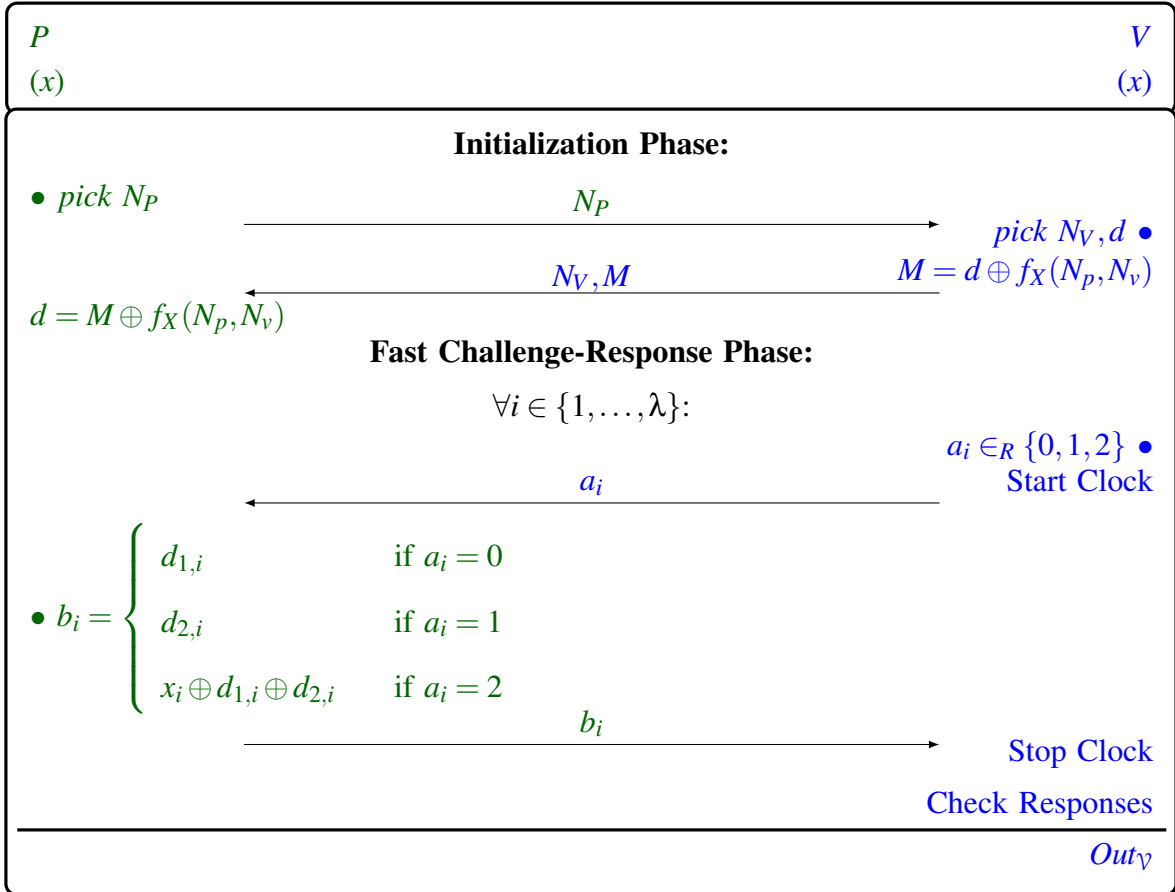


Figure A.4: TDB Protocol with PRF Masking [BMV12]

then she sends a noisy version of the response table to the helper in such a way that the helper passes the noise threshold. In this way, the helper still has a good amount of uncertainty about the secret key of prover.

Čapkun *et al.* [ČBH03]

Bellare-Rogaway [BR93] introduced *mutual-authentication* in which the two participants of the protocol play both of the authentication roles; prover and verifier. Based on their definition, at least a three round conversation is needed to achieve *mutual-authentication*.

This definition brought the attention of researchers of low-power devices to use this concept to make key-agreement protocols [WC01, CH06, BCEP04, JP02, SP07b] and distance-bounding protocols [ČBH03, SP07b, KAK⁺08]. In this literature, there are both symmetric-key and public-key systems. In this section we discuss the existing mutual distance-bounding schemes. Mutual distance-bounding is an extension of classic DB, in a way that both parties do the distance measuring about the other party.

The practical cost of *Mutual-DB* is almost double the cost of classic DB. In classic DB, the main operation at the verifier is time measurement, and the main operation at the prover is fast responding. While in *Mutual-DB*, both parties need to do both of these operations.

Čapkun *et al.* proposed a mutual distance-bounding model. Any pair of parties, who have a shared-key prior the protocol, can run the DB protocol. In this model, both ends of a DB session, prove their authenticity and distance upper bound to each other. They proposed MAD protocol (Figure A.5), which is secure against *DF* and *MF* attacks.

There are two technical challenges in this protocol; first, there is no time gap between challenge-response rounds, which makes the implementation even harder than classic DB protocols, and second, all challenge-response rounds are dependent to the previous rounds, which makes the protocol more fragile in presence of noise.

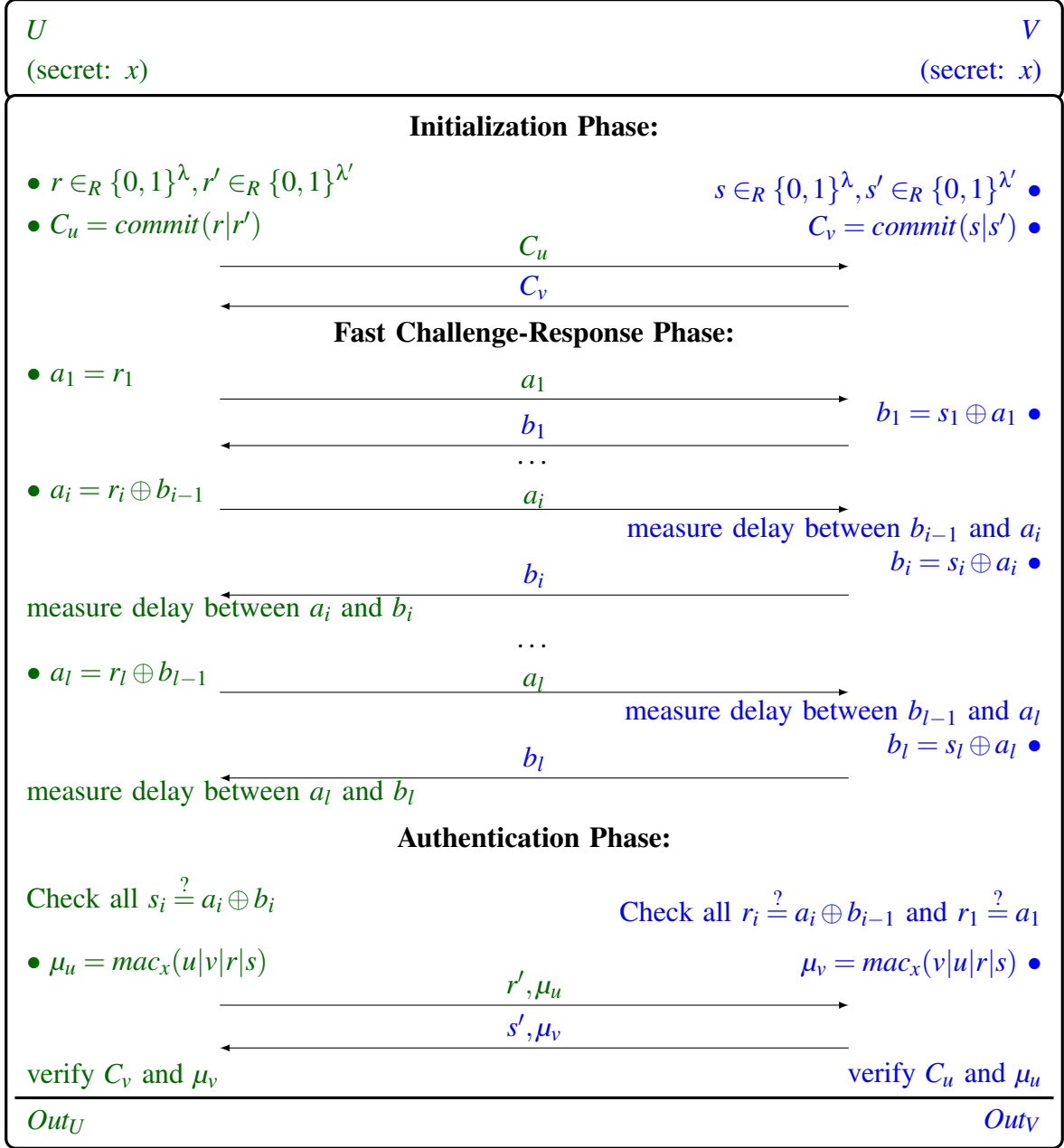


Figure A.5: Mutual Authenticated Distance Bounding protocol (MAD) [ČBH03]

Singelée-Preneel [SP07a]

Singelée-Preneel [SP07a] improved MAD protocol to support noisy channels. This protocol (Figure A.6) uses error correcting code (ECC) and MAC for noise tolerance. However, this protocol inherits the technical difficulties of MAD [ČBH03].

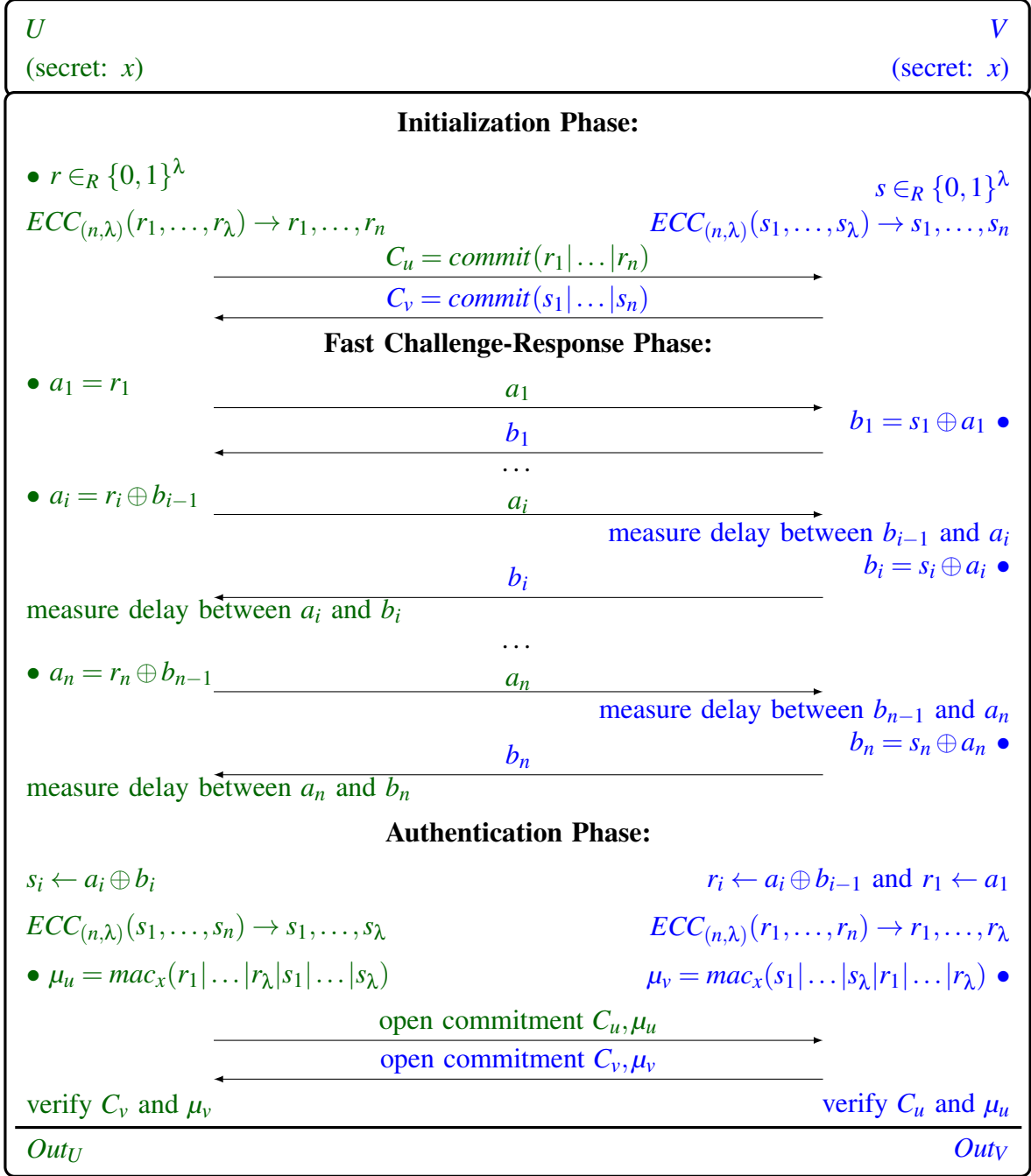


Figure A.6: Mutual Authenticated Distance Bounding protocol [SP07a]

As we discussed earlier, Čapkun *et al.* [ČBH03] designed the first DB protocol with *mutual-authentication*. The parties run the protocol to make sure about the proximity and mutual authentication. Note that *mutual-authentication* is different from *mutual-DB* and has no guarantee about the distance between the parties.

Singelée-Preneel [SP07b] mixed the Diffie-Hellman [DH76] key establishment protocol and MAD DB protocol [ČBH03]. In these two protocols, an ad-hoc network is considered where all parties have the same role. Both sides of the protocol challenge the other side, for DB measurements. In other words, these two works present *mutual-DB* as well as *mutual-authentication*.

Note that the Swiss-Knife protocol (Figure 3.7) [KAK⁺08] modified the Map1 protocol [BR93] in order to construct a classic distance-bounding protocol. However, since there is no mutual distance measurement, then it is not a *mutual-DB* protocol.

A.1.1 Formalizing Security in Distance-Bounding

For the first time in DB literature, Boureanu *et al.* [BMV13b] and Fischlin-Onete [FO13b] proposed formal distance-bounding models. Fischlin-Onete [FO13b] proposed three notions of *terrorist-fraud* attack; GameTF, SimTF and strSimTF, respectively ordered from easy to strong. The latter two models are simulation-based and Fischlin-Onete [FO13a] showed that none of the existing DB protocols offer SimTF, due to the strong definition of SimTF. Fischlin-Onete [FO13b] provided the only SimTF secure scheme by modification of Swiss-Knife protocol [KAK⁺08]. This protocol becomes more vulnerable to other attacks, which may indicate that SimTF security cannot be achieved efficiently.

Boureanu *et al.* [BMV13b] proposed a simpler model, which is trying to be as general as possible to capture all DB attacks. The proposed definitions of *DF* and *MF* are more general than classic definitions, but the proposed *TF* model has a weaker adversary than classic *TF* models. Based on this definition, if there is any PPT adversary who can win the *TF* game with probability γ , then there exist a weaker MiM adversary who can succeed in an specific MiM game with probability γ' .

The proposed SKI protocol [BMV13b] extends the protocol of Avione *et al.* [ALM11]. It works in noisy environments and resists against all DB adversaries. The formal security proof of this protocol is provided in the paper, but despite the claim, this protocol is not proven to be secure

under the defined *TF* adversary. The provided proof is just for deterministic PPT adversaries in the *TF* game, rather than any PPT adversary.

A.2 Implementation of Distance-Bounding

DB protocols make some computational assumptions about the underlying interfaces; (i) messages travel with speed of light, (ii) the parties can send/receive messages that contain a single bit, and (iii) the processing time at the challenged party (*i.e.*, prover) is close to zero. These assumptions allow the high-level DB protocols to simply measure the mutual distance between prover and verifier, by the following formula; $distance = \frac{round\ trip\ time}{speed\ of\ light}$.

There has been many implementations in RFID environments, like keyless entry of cars [Ros98] or [Han11, HK05], which are very restricted because of their abilities and settings, like memory size, processing abilities, channel occupancy status, transmission frequency/rate, or etc. The nice fact about RFID implementations is that there is not many layering in the transmission process (unlike WiFi environment), which makes the time measurements more feasible.

When we want to implement a DB application, we notice that the latter two assumptions are either not true in some environments, or very hard to achieve. That's why the implementation of DB protocols in the existing wireless platforms has been a challenging problem.

As noted in Hancke [Han11], conventional communication channels are designed for reliable data transfer. As a result, these channels feature redundancy and timing tolerances to prevent bit errors. Such latency introduces uncertainty into the distance-bounding measurement and can be exploited by an attacker not adhering to the communication channel to gain a time advantage, which can be used for DB attacks.

As Clulow *et al.* [CHKM06] notes, any DB protocol should consider the following principals to optimize the communication:

- Use a communication medium with a propagation speed that approaches the speed of light.
- Use a communication format in which the recipient can instantly react on the reception of each individual bit. This excludes most traditional byte or block-based communication formats, and in particular any form of redundancy such as error-correction and packet delimiters such as headers and trailers.
- Minimize the length of the symbol used to represent each single bit, or if working with a baseband signal, the verifier should sample as early as possible during the bit period and base his decoding decision on the value of this single sample.

In the distance-bounding literature, there are just a few implementations. The main concern in the implementation of DB protocols, is to reduce the time for receive, process and transmit signal at the prover in order to generate the proper response for the received challenge. This timing should be comparable with propagation time of the signal. There are two general ways for generating the response at the prover; analog processing and digital processing.

The processing time in a digital implementation is consisting of (i) converting the analog symbol of challenge signal to a digital bit, (ii) computing the proper response bit, and (iii) generation and transmission of analog symbol for the response bit. The most efficient digital processing DB implementation is presented in Tippenhauer [Tip12]. As expected, this implementation tries to understand the challenge symbol and process the digital result by using her key bit, and then transmit the response. The timing of this process is about $170ns$. This timing allows the attacker to cheat about $27m$ about the actual distance. Although the timing is not very good in this case, but it supports TF resistance.

However, Rasmussen-Capkun [RC10] presented the fastest analog-processing implementation of DB protocols, which takes $1ns$ for challenge reception, processing and response transmission. This timing provides a tight security guarantee ($15cm$). This implementation cannot support

TF-resistance, because they don't care about the information content of challenge signal. This design is based on a processing method, called Challenge Reflection with Channel Selection (CRCS). This implementation uses three (non-overlapping) communication channels. The verifier sends its challenge bits to the prover using one communication channel (C_0), whereas the prover replies using two communication channels (C_1, C_2) (Figure A.7). While it is receiving the verifier's challenge bit (i.e., the signal that encodes it), the prover is responding with the same signal (bit), but it is sending it on either channel C_1 or channel C_2 , depending on its current input (i.e. key) bit $N_p[i]$.

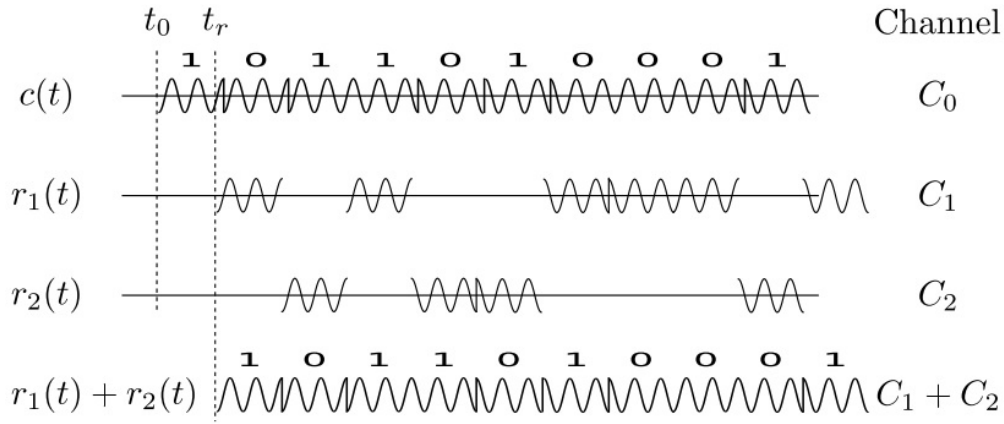


Figure A.7: The verifier measures the time between sending a challenge signal $c(t)$ and receiving the reply signal $r(t) = r_1(t) + r_2(t)$. If $c(t) = r(t)$, the distance bound to the prover is then given by $(t_r - t_0) \cdot c$, where c is the speed of light. [RČ08]

The schematic of prover's circuit is shown in Figure A.8. As it's shown in the figure, the content of the challenge signal is not important to the prover. The prover only makes two copy of the signal, and chooses one of them based on her key bit. Although the timing of this implementation is very good, but it just supports *DF* and *MF* resistance.

Ranganathan *et al.* [RTŠ⁺12] presents an implementation that makes a hybrid digital/analog circuit, called *Switched Challenge Reflector with Carrier Switching* (SCRCS), which enables the implementation of *TF*-resistant DB protocols. In this implementation, the allowed *TF* inaccuracy is reduces to 4.5m (30ns).

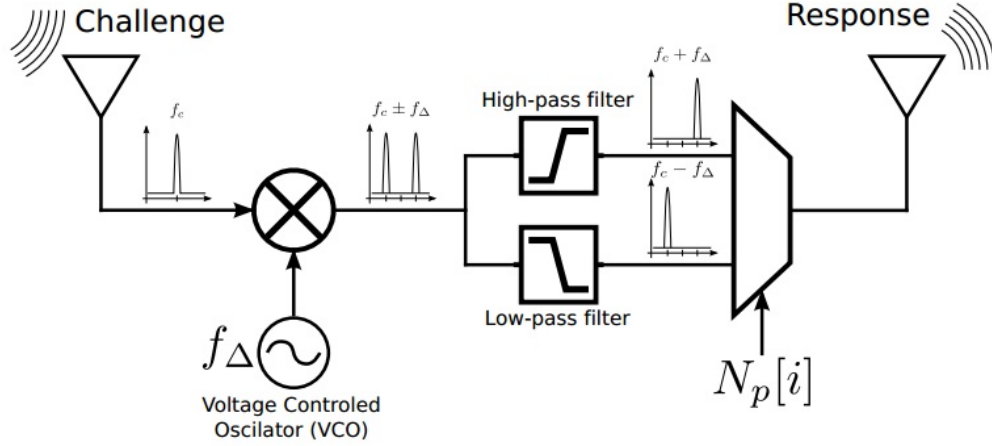


Figure A.8: Schematic of prover by using CRCS [RČ08]

The design of SCRCs is shown in Figure A.9, Figure A.10, Figure A.11 and Figure A.12.

In this design, the verifier transmits challenges on one of the two different carrier frequencies and the prover duplicates the received signal into two different minor shifted carriers, and based on her key bit, replies with one of the two copies. Note that the verifier and prover are synchronized at the beginning of each round, which can happen by transmitting some preamble at the beginning of each round. It's important to know that the length of the preamble and the processing related to that, does not effect the distance-bounding security implications. Four possible reply channels are created before activating the appropriate reflected carrier frequency. The verifier listens on the expected channel (among four channels), and checks the time delay between the transmitted and the received signal. This implementation allows *DF*, *MF* and *TF* resistance and at the same time, the timing of prover's operations is about $30ns$, which allows only $4.5m$ distance cheating in *TF* and $41cm$ distance cheating in *DF* and *TF* scenarios.

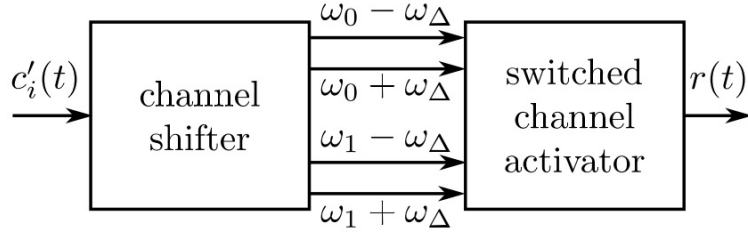


Figure A.9: Overview of the switched challenge reflector with carrier shifting [RTŠ⁺12]

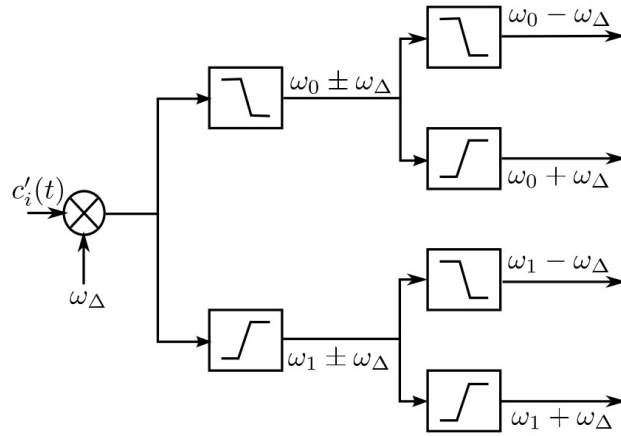


Figure A.10: The channel shifter. The incoming signal $c'(t)$ contains the challenges on either carrier frequency w_0 or w_1 . After mixing $c(t)$ with w_Δ , the signal is filtered appropriately to generate the four possible response channels: $w_0 - w_\Delta$, $w_0 + w_\Delta$, $w_1 - w_\Delta$, $w_1 + w_\Delta$. [RTŠ⁺12]

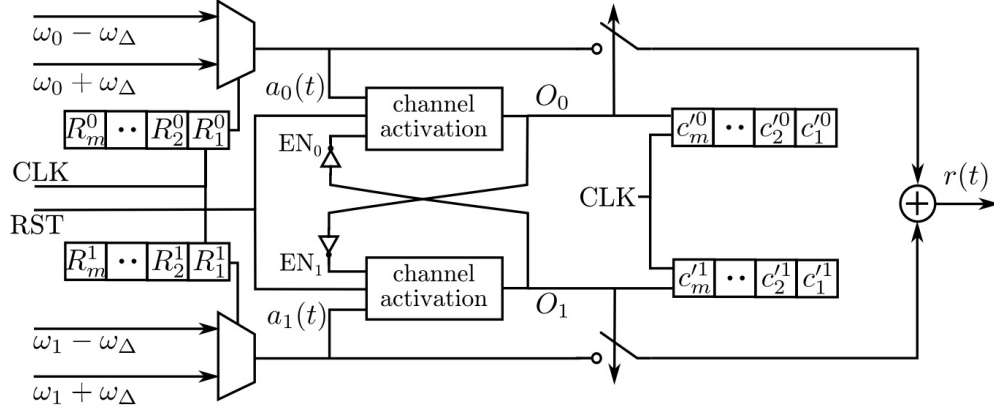


Figure A.11: Switched channel activator. The registers R^0 and R^1 , which are derived from the key of prover, select which two of the four reply channels are used in this round. The channel in which sufficient energy is encountered first gets enabled. After a channel is activated, it stays active until the end of this rapid bit-exchange round while the other channels remain deactivated until the end of this round. [RTŠ⁺12]

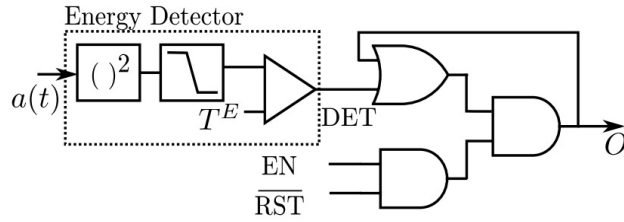


Figure A.12: Internals of channel activation. We obtain a DC component of the squared signal to detect energy in the channel and store the value for this round in a latch-like circuit. In summary, the “Energy Detector” returns 1 if there is a signal on the input, and 0 otherwise. The channel activation can be disabled by pulling EN (enable signal) low and is automatically reset at the beginning of each round of the rapid-bit exchange (RST). [RTŠ⁺12]

Ranganathan et al. [RTŠ⁺12] mentioned a new attack model that can target the analog-processing implementations. In “Double Read-out” attack, the attacker tries to simultaneously query both of the registers (i.e. both possible responses of prover), which usually ends up in leaking the secret key of prover. The analog implementations (e.g. CRCS [RČ08]) are typically vulnerable to this

attack, since they allow the challenge signal on both possible carrier frequencies at the same time.

A.3 Privacy in Distance-Bounding

In this section, privacy of a prover is considered as un-traceability of different sessions of the prover. We can consider two different cases in this scope;

- *Pseudonymity*: There is no PPT adversary \mathcal{A}_P , with a certain view and certain capabilities, who can associate a DB session of a prover \mathcal{P}_i to the registration session of the same prover, better than a random guess.
- *Unlinkability*: There is no PPT adversary \mathcal{A}_U , with a certain view and certain capabilities, who can correlate any two different DB sessions of a single prover, better than a random guess.

In general, both cases have been considered together as privacy. Obviously the important factors in this definition is the view and capabilities of the adversary, which makes different privacy models.

This problem has been studied against Man-In-the-Middle (MiM) adversaries ([Vau07], [HPVP11]), while verifiers are trusted. In these systems the location of provers are known by the verifiers, but provers hide their identity in their interactions. However, since these works are on RFID platforms, which need low computation cost, the existing protocols and privacy models are considering symmetric-key structures.

Moreover, in a symmetric-key DB protocol, there is a shared-key between prover and registration authority and so privacy is not achievable against an adversary who can have access the internal state of the authority. One can always achieve prover privacy by giving the same key to all provers. This however is unacceptable because of the high compromisation risk of the whole system.

Defining privacy in authentication protocols has attracted much research [Vau07, PV08, NSMSN08, JW09, MLDL09, DLYZ11, HPVP11, HPO13, GOR14]. In the following we give an overview of

the related works.

Vaudenay [Vau07]

Vaudenay [Vau07] proposed one of the most general MiM privacy models in RFID communication setting. In this model, the privacy adversary is a MiM attacker (\mathcal{A}) between verifier and prover. \mathcal{A} can block the connection between the two parties and communicate with them separately. In RFID protocols, there are multiple tags and readers, and they are distributed in many locations. The security model of RFID protocols are similar to the security model of DB protocols.

In this model, the interactions of \mathcal{A} is modeled with access to some oracles in the security model. Other than the initiation and communication abilities of \mathcal{A} , there are two special oracles as follows; **Result** oracle gives the final result of a verification session to the adversary, and **Corrupt** oracle returns the non-volatile internal state (i.e. secret key) of a prover to the adversary. In this model, the adversary is willing to distinguish between the sessions of two provers.

In this model the adversaries are classified, based on their access to these two oracles. A *wide* adversary has access to **Result** oracle, otherwise it will be a *narrow* adversary. In parallel, the adversary can be *weak* (no access to **Corrupt** oracle), *forward* (**Corrupt** queries can only be followed by other **Corrupt** queries), *destructive* (**Corrupt** queries destroys the access to the corrupted prover), and *strong* (unlimited access to **Corrupt** oracle).

Therefore, the following order in the classification holds; $narrow \subseteq wide$, based on having access to **Result** oracle. And $weak \subseteq forward \subseteq destructive \subseteq strong$ based on having access to the state of **Corrupt** oracle. In this model, a simulation-based definition about privacy is considered, which is too strong and Vaudenay proofs that it's impossible to achieve the defined *strong* privacy.

Hermans *et al.* [HPVP11]

Hermans *et al.* presented a simpler game-based privacy model of MiM attacks. This work is an extension of Vaudenay [Vau07]. In this model, privacy is defined as a game between an *adversary* and a *challenger*. In this game, there are two provers, and the challenger chooses one of them randomly and allows it to run the protocol. The adversary is willing to find out which one has been chosen by the *challenger*. The *adversary* has access to the same oracles as in Vaudenay's [Vau07] model.

Peeters-Hermans [PH12] added a new oracle to the list of oracles in Hermans *et al.* [HPVP11], by which the adversary can create insider prover and have control on it (`CreateInsider`). The insider provers cannot be chosen by the challenger.

Hermans *et al.* [HPVP11] proved that the public-key RFID protocol of Vaudenay [Vau07] is *wide-strong* private, if it uses an *IND-CCA2* crypto-system. And also proved that the random-oracle protocol of Vaudenay [Vau07] is *narrow-destructive* private.

Hermans *et al.* [HPO13]

Hermans *et al.* designed a public-key DB protocol, which is *wide-forward-insider* or *narrow-strong* private in the privacy model of Hermans *et al.* [HPVP11]. This protocol is also secure in *DF* and *MF* DB models. However, this protocol is trivially vulnerable to *TF*, which makes it hard to be adopted for payment systems or etc.

Gambs *et al.* [GOR14]

Since Paise-Vaudenay [PV08] showed that *destructive* and *strong* privacy is not achievable in symmetric-key systems. For the first time in the literature, Gambs *et al.* [GOR14] decided to modify Hermans *et al.* [HPO13] protocol, in order to go beyond MiM privacy adversaries. The proposed privacy model, protects the privacy of provers against *malicious* verifiers or *honest-but-curious* registration authority.

They extended the protocol of Hermans *et al.* [HPO13] to propose a public-key DB protocol that is privacy-preserving. This protocol is designed to be secure in DF and MF DB models and privacy-preserving against three separate adversaries: (1) MiM adversary, (2) a malicious verifier, and (3) an honest-but-curious adversary who knows the internal state of the verifier and the registration authority.

Unfortunately, this protocol is broken by a simple MiM attack. However, although this is a stronger form of privacy (than MiM privacy models), but since the protocol provides deniability to provers, then it will be hard to be used in payment systems. Moreover, since this protocol is an extension of [HPO13], then it inherits the trivial TF attack, which makes it even harder to be used in a payment system. This protocol requires to update the public-key of all provers dynamically when a new prover is added or removed from the system, which makes the protocol to have high cost per operation.