Proof of Proximity of Knowledge

Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasec.epfl.ch

Abstract. Public-key distance bounding schemes are needed to defeat relay attacks in payment systems. So far, only two such schemes exist, but fail to fully protect against malicious provers. In this paper, we solve this problem. We provide a full formalism to define the proof of proximity of knowledge (PoPoK). Protocols should succeed if and only if a prover holding a secret is within the proximity of the verifier. Like proofs of knowledge, these protocols must satisfy completeness, soundness (protection for the honest verifier), and security (protection for the honest prover). We construct ProProx, the very first fully secure PoPoK.

1 Introduction

Following the chess grandmaster problem, a dummy chess player can challenge two grandmasters in parallel (without letting them know) if they take different colors. This works by letting the dummy player relay the move of a grandmaster on one chess board to the other chess board against the other grandmaster and *vice versa*. In real life, *relay attacks* can be a serious threat against applications such as NFC-based payment: for small payments, there is typically no action required on the creditcard or smartphone (beyond approaching to the terminal) such as typing a PIN code. So, a man-in-the-middle adversary could just relay communications between the payment device of the victim and the terminal to make payments on the behalf of the holder. The limit of the speed of communication was proposed to solve this problem [4]. Brands and Chaum [12] introduced the notion of *distance-bounding protocol* to prove that a *prover* is close enough to a *verifier*. This relies on information being local and unable to travel faster than the speed of light. So, an RFID reader can identify when participants are close enough because the round-trip communication time in challenge/response rounds have been small enough. Although challenging from an engineering viewpoint, it is actually feasible to implement these protocols, as shown in [25,26,27]. Actually, some commercial NFC cards by NXP[1] are already supposed to implement "proximity checks against relay attacks" by "a precise time measurement of challenge-response pairs", although no public analysis is available so far.

The literature considers several threat models.

- *Relay attack*: an adversary relay messages between a far-away honest prover and a verifier, trying to make the verifier accept. This is extended by *Mafia fraud* [16] where the adversary can also modify messages. This is further extended by *Man-in-the-Middle attack* [6,8] where the attack follows a learning phase where the prover could be close-by. In *Impersonation fraud* [2], the prover is absent and the adversary tries to impersonate the prover to the verifier.
- *Distance fraud* [12]: a far-away malicious prover tries to pass the protocol.
- *Terrorist fraud* [16]: a far-away malicious prover, with the help of an adversary, tries to make the verifier accept the adversary's rounds on his behalf, but without giving the adversary any advantage to later pass the protocol alone. This is extended by *Collusion fraud* [6,8] where the goal of the adversary is now to run a man-in-the-middle attack. Terrorist fraud is also related to

[1] See Mifare Plus X on http://www.nxp.com

the notion of *soundness* [28]: whenever the verifier accepts, there must be an extractor who can reconstruct the secret of the prover based on the view of all close-by participants, even after many iterations. An hybrid model between distance fraud and terrorist fraud is the one of *Distance hijacking* [15]: A far-away prover takes advantage of some honest, active provers to make the verifier accept.

One of the first model to capture these notions was proposed by Avoine *et al.* [1]. However, is was not formal enough. Then, two parallel models were developed: the BMV model by Boureanu *et al.* [6,8] and the DFKO model by Dürholz *et al.* [17]. There exist many (symmetric) distance-bounding protocols but so far only the SKI protocol [5,6,7] (based on the BMV model), the Fischlin-Onete protocol [19] (based on the DFKO model), and DB1 and DB2 [9,10] (combining both protocols in the BMV model) provide an all-encompassing proven security, i.e., they protect against man-in-the-middle, distance fraud, and collusion fraud with a formal proof of security. (See [5, Section 2].)

As discussed herein, resistance to malicious provers (i.e., distance fraud, distance hijacking, terrorist fraud, collusion fraud, and non-soundness) is important, specially for non-repudiation. This is needed for contactless payment applications: If a customer can deny a payment by proving he was somewhere else, there is a liability issue. Making the scheme such that this type of fraud is impossible would certainly be an asset. It would strengthen trust and acceptance of this technology. So, there is a need for distance-bounding protocols protecting against all the above types of fraud.

Public-key distance bounding. In interactive proofs, the prover does not share a secret key with the verifier. The verifier only knows a public key. However, so far, only the following distance-bounding protocols are in the public key model: the Brands-Chaum protocol [12] (Fig. 4), the Bussard-Bagga protocol [13], and the Hermans-Peeters-Onete (HPO) protocol [23] (Fig. 5).[2] The Bussard-Bagga protocol was broken by Bay et al. [3] and none of the others protect against terrorist fraud. Additionally, the protocol VSSDB was presented at the BalkanCryptSec'14 conference by Gambs *et al.*. It is based on the random oracle model, but the security proofs are still questionable, as discussed in Appendix C. So, the problem of making a fully secure public-key distance-bounding protocol is still open.

Sometimes, secret key protocols are sufficient for practical applications. Indeed, to unlock a car or to open a door in a building with an RFID token, it may be reasonable (especially if symmetric distance-bounding protocols are more efficient) to assume that the RFID token shares his secret with the car or the door. However, when considering wireless payment systems, it is unreasonable to assume that the secret of the prover can be shared with all verifiers. When paying with an NFC credit card at the grocery shop, the cashier may sometimes be offline with the bank and not willing to stop business. To protect against relay attacks, we need the cashier to run a distance-bounding protocol from the public key of the payer only. So, there is a need for public-key distance bounding.

Contribution. In clear, our contributions in this paper are as follows.

- We take the "Fundamental Lemma" in its version of [9,10] and provide a detailed proof of it in the BMV model. (The previous version of this lemma was proven in [6].)
- We adapt the framework of [9,10] in the BMV model to provide a full formalization of public-key distance-bounding. We specify our new primitive: the *proof of proximity of knowledge (PoPoK)*. It is based on a prover holding a secret x authenticating to a verifier knowing his public key y. Under honest execution, the verifier should accept if the prover is at a distance up to a bound B. Under adversarial environment, he should only accept if we can extract x from the view of all participants within a distance lower than B.

[2] A variant of the HPO protocol offers anonymous authentication [20].

- We change the definition of soundness from [9,10] and [28] to make it more natural and closer to the one of interactive proofs. So, our model is pretty natural and nicely connects recent work on distance bounding (such as the BMV model [6,8]) and interactive proofs.
- Observing that none of the existing protocol is provably sound, we construct ProProx, the very first secure and sound PoPoK. It is based on the quadratic residuosity problem, using the Goldwasser-Micali encryption [21,22] as a perfectly binding commitment and the Fiat-Shamir protocol [18].
- We provide a technique to prove security and soundness. Essentially, we construct an extractor based on the "Fundamental Lemma" and prove that the protocol is zero-knowledge. Then, the extractor is used with malicious provers to show that the protocol is sound and the extractor with non-concurrent honest provers is transformed into a key recovery algorithm by using zero-knowledge.
- By closely looking at our security proofs, we provide some concrete parameters to provide concrete security and soundness. Our best parameters require the encryption of 80 bits, the ZK proof that 80 residues are quadratic, and 80 rounds of fast challenge/response exchanges. This provides an 80-bit equivalent offline security and an online security which guaranties that the probability of success of online attacks is between 2^{-11} and 2^{-22}. We further conjecture that 80 rounds is optimal for protocols with a single-session extractor.

Organization. In Section 2 we provide a full formal model for public-key distance bounding, mostly based on the BMV model, and define the PoPoK. We relate this to the state of the art and to the notion of interactive proofs. Some case studies illustrating the difficulty of constructing a PoPoK are put in Section 3. In Section 4 we propose ProProx, the very first fully secure PoPoK, and analyze it. In addition to man-in-the-middle and collusion frauds, we specifically cover distance frauds. We also discuss on concrete parameters and implementation.

2 Model and Definitions

We refine the security definitions and other tools from the BMV model [6,8,28]. Throughout this paper, we use the asymptotic complexity approach for security. Constructions depend on some security parameter λ which is omitted for more readability. A *constant* does not depend on λ, while parameters defining cryptographic constructions do. Algorithms will often run in probabilistic polynomial-time (PPT) in terms of λ. A real function $f(\lambda)$ is negligible if for any d, we have $f(\lambda) = O(\lambda^{-d})$, as $\lambda \to +\infty$. In this case we denote $f(\lambda) = \mathsf{negl}(\lambda)$. We also use the following notation:

$$\mathsf{Tail}(n,\tau,\rho) = \sum_{i=\tau}^{n} \binom{n}{i} \rho^i (1-\rho)^{n-i}$$

2.1 Computational, Communication, and Adversarial Models

In our settings, participants are interactive Turing machines running PPT algorithms.

We follow the BMV communication model and adversarial model [6,8]: we assume that participants have a *location* which is an element of a metric space $\mathcal{S}$, with a distance function d. If a participant π_1 at a location loc_1 executes a special command $\mathsf{send}(\pi_2, m)$ at time t to send a message m to a participant π_2 at location loc_2, the message m is received by π_2 at time $t + d(\mathsf{loc}_1, \mathsf{loc}_2)$. Furthermore, any malicious participant π_3 at some location loc_3 could see this message m at time $t + d(\mathsf{loc}_1, \mathsf{loc}_3)$. We assume no authentication: π_2 does not know if the message comes from π_1.

There is an exception preventing m from being delivered to π_2: if π_2 is honest and some (malicious) participant π_3 at some location loc_3 has sent a special signal $\mathsf{corrupt}(\pi_1, \pi_2)$ at time t', m is not

delivered to π_2 if $t + d(\mathsf{loc}_1, \mathsf{loc}_2) \geq t' + d(\mathsf{loc}_3, \mathsf{loc}_2)$. This condition is a consequence of the information traveling with a speed limit: whenever a malicious participant π_3 corrupts a $\pi_1 \to \pi_2$ channel, π_2 will only receive the messages until his corruption signal emitted from π_3 reaches π_2.

Note that once the $\pi_1 \to \pi_2$ channel is corrupted, π_3 can still see the message m sent by π_1 and decide to send any m' to π_2, either depending on m if he waits to receive m, or not. The crux is that either m' is independent of m, or it is delivered at a different time, depending on the locations.

The BMV model is only used to prove the following lemma which is taken from [9,10].

Lemma 1 (Fundamental Lemma). *Assume a multiparty protocol execution with a distinguished participant $\mathcal{V}$, the set* Far *of all participants within a distance to $\mathcal{V}$ larger than B, and the set* Close *of all other participants. At some time t in the execution, $\mathcal{V}$ broadcasts a message c and waits for a response r. We let* Acc *denote the event that r was received by $\mathcal{V}$ no later than time $t + 2B$. For each participant U, we denote by* View_U *his partial view just before time $t + d(\mathcal{V}, U)$, the time when U could see c. We say that a message from U is* independent[3] *(from c) if it is sent before time $t + d(\mathcal{V}, U)$: if it is the result of applying the algorithm U on* View_U*, or on an earlier partial view. There exists an algorithm* Algo *such that if* Acc *occurs and r was sent by some $U \in$* Close*, then* $r = \mathsf{Algo}(\mathsf{View}_{\mathsf{Close}}, c, \mathsf{Other})$*, where* $\mathsf{View}_{\mathsf{Close}}$ *is the list of all* View_C*, $C \in$* Close*, and* Other *is the list of all messages from any $F \in$* Far *which are independent from c (i.e., sent before time $t + d(\mathcal{V}, F)$), that at least one member $C \in$* Close *could see (either because he was the recipient, or because he is malicious), and not already in any* View_C *(i.e., sent after time $t + d(\mathcal{V}, C) - d(C, F)$). Furthermore, if* Acc *occurs and r was sent by some $U \in$* Far*, then r is independent from c.*

In clear: r cannot depend on messages from a far away U which has been sent after U received c.

We provide here a detailed proof of this lemma in the BMV model.

Proof. The case where r comes from $U \in \mathsf{Far}$ is easy: if r is computed at time t', we must have $t' \leq t + 2B - d(\mathcal{V}, U)$. Since $d(\mathcal{V}, U) > B$, we have $t' < t + d(\mathcal{V}, U)$, so r is independent form c.

For the other case, we let Algo simulate each participant $C \in \mathsf{Close}$ between time $t + d(\mathcal{V}, C)$ (to continue after View_C) and time $t + 2B - d(\mathcal{V}, C)$ (after which it is too late to send a message to $\mathcal{V}$). The simulation of all participants is done in parallel, chronologically. The output of Algo is the first message r for $\mathcal{V}$ to arrive to $\mathcal{V}$.

We show by induction that the simulation cannot block: when Algo must simulate the computation of a message by some $C \in \mathsf{Close}$ at time $t' \leq t + 2B - d(\mathcal{V}, C)$, the input to C are messages m which are either part of the input to Algo or which have been computed by Algo before. To prove this, we distinguish three cases. First, if m comes from a participant in Far, due to the distance constraint, it must be independent from c, so be part of Other which is input to Algo. In the second case, m comes from $\mathcal{V}$. Since it is assumed that $\mathcal{V}$ does not send messages after c before r is received, m is either part of View_C or the message c itself, so part of the inputs of Algo in both cases. In the third case, m comes from $C' \in \mathsf{Close}$. It is either computable from $\mathsf{View}_{C'}$ or computed by C' after $\mathsf{View}_{C'}$ was closed but early enough to arrive at C at time t', so between time $t + d(\mathcal{V}, C')$ and $t' - d(C, C')$. Since $t' \leq t + 2B - d(\mathcal{V}, C)$, we have $t' - d(C, C') \leq t + 2B - d(\mathcal{V}, C) - d(C, C') \leq t + 2B - d(\mathcal{V}, C')$, so this message must have been computed in the simulation by Algo.

Finally, we observe that Algo returns r. Indeed, the message r to $\mathcal{V}$ must be among those which are computed by the simulation, due to the distance constraints. □

Participants can move, in a way which can depend on their view, but not faster than communication. I.e., the location at time $t + 1$ can only be at a distance up to 1 to the location at time t. For

[3] We stress that this notion of independence is *not* related to statistical independence. Our notion of independence means that the message is computed with data available before c has reached the location where the computation is done.

simplicity, we assume that far-away participants (as defined in Def. 3) remain far away during the entire execution.

We sometimes consider that when an honest participant receives a message from another honest participant, it may be subject to noise. As for malicious participants, we could assume that they use a better equipment which eliminates noise. Also: whenever the honest-to-honest communication is not time-sensitive, we may also assume that they use error correction means so that the communication is noiseless. Protocols shall thus make clear which messages may be subject to noise.

2.2 PoPoK: Proofs of Proximity of Knowledge

Definition 2 (Proof of proximity of knowledge). *A proof of proximity of knowledge (PoPoK) is a tuple* $(\mathcal{K}, \mathsf{Kgen}, P, V, B)$*, consisting of: a key space* $\mathcal{K}$ *depending on a security parameter* λ*, with elements of polynomially-bounded size in terms of* λ*; a PPT algorithm* Kgen*; a two-party PPT protocol* $(P(x), V(y))$*, where* $P(x)$ *is the* proving algorithm *and* $V(y)$ *is the* verifying algorithm*; a distance bound* B*. The algorithm* Kgen *maps the secret* $x \in \mathcal{K}$ *to a public key* y*.*[4] y *is given as input to all participants. At the end of the protocol,* $V(y)$ *sends a final message* Out_V*. He accepts (*$\mathsf{Out}_V = 1$*) or rejects (*$\mathsf{Out}_V = 0$*).*

The protocol must be such that when running $P(x)$ *and* $V(y)$ *at locations within a distance up to* B*, in a noiseless environment, the verifier always accepts. This property is called* completeness*.*

If the protocol specifies a list of time-critical challenge/response exchanges, we say that it is complete with noise probability p_{noise} *if, in an environment in which all challenge/response rounds are independently corrupted with probability* p_{noise} *and other exchanges are not subject to noise, the probability that the verifier accepts is* $1 - \mathsf{negl}(\lambda)$*.*

The last part of this definition applies to protocols which identify well the challenge/response rounds.

We implicitly assume that (x, y) keys of honest provers are correctly registered using Kgen.

In practice, if we want to have $B = 10$m, assuming that an adversary can do computation in negligible time, the timer for receiving a response r to a challenge c in Lemma 1 should be limited to 67ns. So, an honest prover at a zero distance must respond within less than 67ns. This clearly excludes any cryptographic computation. This even excludes waiting for receiving several bits in a sequence since the typical time between two bits is of 1μs in wireless technologies. To be realistic, a PoPoK can only consider boolean (or very small) challenges and responses when it comes to use Lemma 1.

2.3 Cryptographic Properties of PoPoK in a Multiparty Setting

We adopt the multiparty setting from [9,10] and only adapt it to accommodate public-key distance bounding.

As it is discussed in Example 7, we don't assume that instances reliably know their locations.

We consider a setting with participants which are called either *provers*, *verifiers*, or *other actors*. We assume only one verifier $\mathcal{V}$, without loss of generality. Indeed, we can take other verifiers as *other actors*. Similarly, we assume that provers correspond to the same identity so share the same secret x, without loss of generality. (Provers with other secrets are considered as *other actors*.) Other actors are malicious without loss of generality. The difference between malicious provers and malicious actors is in the input: malicious provers receive x while malicious actors only receive y.

[4] This is without loss of generality: an algorithm $\mathsf{Gen}(\rho) = (K_p, K_s)$ making a public key K_p and a secret key K_s using coins ρ defines such $\mathsf{Kgen}(x) = y$ by letting $x = \rho$ and $y = K_p$.

We assume that participants run their algorithm only once. Since different participants may correspond to the same identity at different time and locations, multiple executions are modeled by multiple participants. Since honest provers correspond to the same person at different time and locations, we assume that honest provers never run concurrently. A malicious prover may however clone himself at different locations and run algorithms concurrently.

Definition 3 (Experiment). *An* experiment exp *for a PoPoK* $(\mathcal{K}, \mathsf{Kgen}, P, V, B)$ *is defined by several* participants *who are a* verifier $\mathcal{V}$*,* provers *from an ordered set* $\mathbf{P}$*, and other* actors *from a set* $\mathbf{A}$*. Participants which are within a distance of at most B to* $\mathcal{V}$ *are called* close-by *participants. Participants which are within a distance larger than B to* $\mathcal{V}$ *are called* far-away *participants. We say that the prover is* always far-away *if all participants in* $\mathbf{P}$ *are far away.*

We adopt a static *adversarial model: either the prover is honest and all participants in* $\mathbf{P}$ *run the* $P(x)$ *algorithm, or the prover is malicious, and they could run any PPT algorithm.*

If the prover is honest, the participants in $\mathbf{P}$ *are assumed to be* non-concurrent*: at each time, there is only one participant of* $\mathbf{P}$ *which is activated. When the active participant terminates, another participant of* $\mathbf{P}$ *is activated, following a sequence. This sequence is defined by the ordering of* $\mathbf{P}$*.*

At the beginning of the experiment, for malicious provers, (x, y) *is set arbitrarily. If the provers are honest,* $x \in \mathcal{K}$ *is randomly selected and* $y = \mathsf{Kgen}(x)$ *is computed. Then, x is given as input to participants in* $\mathbf{P}$*, while y is given as input to all participants.* $\mathcal{V}$ *runs* $V(y)$*. All participants are then activated and run concurrently. (If the prover is honest, only the first prover from* $\mathbf{P}$ *is activated with the other participants, other provers being activated sequentially.) The experiment terminates when* $\mathcal{V}$ *produces its final output* Out_V*.*

We want to protect the honest prover against abuse. For that, we formalize security as follows:

Definition 4 (Security of PoPoK). *A PoPoK* $(\mathcal{K}, \mathsf{Kgen}, P, V, B)$ *is* secure *if for any experiment* exp *where the prover is honest and always far-away from* $\mathcal{V}$*, we have* $\Pr[\mathsf{Out}_V = 1] = \mathsf{negl}(\lambda)$*.*

This definition clearly captures relay attacks, Mafia fraud [16], and man-in-the-middle attacks in general. Some models (like in [6,8]) distinguish a learning phase (with provers which could be close-by) and an attack phase (with far-away provers). This could also be described in the above framework: we just simulate both phases in two different locations and time of the experiment and make the adversary of the (simulated) learning phase sends what he has learnt to the adversary of the (simulated) attack phase. (The target $\mathcal{V}$ would be the one from the attack phase. So, provers would still be all far away.) With our model, we also cover more general cases in which provers are being attacked in parallel at other locations. Our definition is thus much simpler, more natural, and more general than [6,8].

We now formalize the protection for the honest verifier. Intuitively, we want that if the proof is accepted, it must be because the information about the secret x is in the close-by neighborhood.

Definition 5 (Soundness of PoPoK). *A PoPoK* $(\mathcal{K}, \mathsf{Kgen}, P, V, B)$ *is* sound *if there exists a negligible function* $p_{\mathsf{Sound}}(\lambda)$ *such that for any experiment* exp *in which* $\mathcal{V}$ *accepts with probability* $\Pr[\mathsf{Succ}] \geq p_{\mathsf{Sound}}(\lambda)$*, there exists an algorithm* $\mathcal{E}$ *called* extractor*, with the following property.* exp *defines an oracle which simulates an execution of* exp *and returns the views of all participants which are close-by (excluding* $\mathcal{V}$*) and the transcript of the protocol seen by* $\mathcal{V}$*.* $\mathcal{E}$ *can invoke the oracle many times. Then,* $\mathcal{E}$ *finally outputs* x' *such that* $\mathsf{Kgen}(x') = y$*, using an expected time complexity of* $\frac{\mathsf{Poly}(\lambda)}{\Pr[\mathsf{Succ}]}$*.*

For experiments with a close-by prover, this is trivial (as x is in the view of the prover). For experiments with no close-by prover, close-by actors would extract the prover's credential in the case the verifier would accept due to the prover cheating from far away. For experiments with no close-by participant at all, the transcript as seen by $\mathcal{V}$ would leak. So, a malicious prover is bound to leak.

Compared to the soundness of interactive proofs, our notion uses a straightline extractor: we extract the secret from close-by participants without rewinding them. This makes the treatment of multiparty settings much easier. As we will see, our extractor essentially uses Lemma 1. Interestingly, the extractor is also used to prove security: if the protocol is zero-knowledge, the oracle extractor can (in honest prover cases) be transformed into a stand-alone extractor which contradict some one-way assumption in the key generation.

Clearly, our definition nicely connects the infamous terrorist-fraud resistance to the soundness of interactive proofs. To compare with the literature, we could see that terrorist frauds in our model make the secret leak instead of only making man-in-the-middle attack feasible as in the notion of collusion fraud proposed in [6,8], and on which the SKI protocol is based, or only making impersonation attack feasible as in [9,10]. Our soundness is thus stronger.

Our notion and the one of [9,10] are close to soundness as defined in [28], except that we no longer require $1/\Pr[\mathsf{Succ}]$ to be polynomial. Also, compared to [9,10], we no longer need the condition on the success of the experiment to extract and we call an oracle O many times instead of using m views.

Just like other notions of TF-resistance, soundness is incomparable with SimTF-security [17] or GameTF-security [19] in the DFKO model. But SimTF-security has a single symmetric instance [19] which is not competitive (see [9,10,28]) and GameTF has no instance (the MSK instance from [19] is broken in [28]).[5]

What is captured by the notion of soundness? In distance bounding, the regular distance fraud is the case where the verifier accepts, with a malicious prover, and all participants far-away. Our soundness definition implies in such a case an extractor based on the transcript only. So, our soundness and security definitions protect against distance fraud by warning that any attack would leak. If the malicious prover does not care about leaking, this line of defense does not work and we have to treat distance fraud specifically. For this reason, we specifically define resistance to distance fraud in Def. 6 below.

Similarly, distance hijacking is a more general case allowing some honest close-by participants but not the prover holding x. We could say that a malicious prover hijacking the distance is bound to leak his secret due to our soundness definition. Indeed, the distance hijacking scenario is less general than the case of terrorist fraud where some adversaries may be close to the verifier.

Why do we care about soundness? In practice, it is not quite clear which application *really* needs soundness. Indeed, the original motivation of distance-bounding was to protect against relay attacks, or more generally to man-in-the-middle attacks, where the prover is assumed to be honest. These are all covered by security following Def. 4. When we additionally want to protect against malicious provers, one may think that distance fraud resistance it is enough and not care about soundness.

History shows that more bizarre attack scenarios happen, such as distance hijacking and collusion fraud. As we have seen, our notion of soundness is strong enough to capture all these threat models. Finally, protection against malicious provers in general (what soundness captures) is really needed for applications requiring non-repudiation such as payment. A malicious prover who succeeds to make a payment by an unknown attack can deny having made the payment with an alibi for having been unable to do it honestly (e.g., by proving he could not be close to $\mathcal{V}$ at this time). Soundness could offer more confidence in that this situation could not happen. It could be an asset.

We mention that there are other techniques to protect against malicious provers, including making sure that provers are honest by relying on tamper resistance. But tamper resistant devices are always subject to side-channel attacks, so this may not be so easy to achieve. Also, we think that soundness nicely bridges to the theory of interactive proofs of knowledge where soundness is defined similarly.

[5] The VSSDB case is discussed in Appendix C.

There is however a risk in trying to make distance bounding sound. Like in [23], some authors argue that soundness weakens protocols. This is actually the case of [19] (as shown in [28]). Also, sometimes, poorly analyzed protocols aimed to be sound can be badly broken, like DBPK-Log [13] (as shown in [3]) or MSK [19] (see [28]). This requires soundness to be taken with a special care and to make correct full security proofs.

Definition 6 (Distance-fraud security). *A PoPoK* $(\mathcal{K}, \mathsf{Kgen}, P, V, B)$ *resists to* distance fraud *if for any experiment* exp *where all participants are far away from* $\mathcal{V}$*, we have that* $\Pr[\mathsf{Out}_V = 1] = \mathsf{negl}(\lambda)$.

3 Case Studies

We briefly discussed some natural examples which come to mind and which illustrate that making a protocol which is both secure and sound is not trivial.

Example 7. Imagine a protocol in which the prover signs his location (with a challenge), then the verifier checks if this location is close enough. Obviously, this protocol is not sound since a malicious prover could claim to be anywhere. It may not be secure either *in practice*, since an adversary could make the honest prover believe that he is somewhere else. He could just relay GPS, WiFi, or cellular network signals. So, we rather consider protocols not assuming they reliably know their location.

Example 8. We consider a very simple protocol in which a verifier sends a random bitstring as a challenge and the prover must return at once a valid signature of it. This protocol suffers from many problems. First of all, it may not be sound, except if we could make the signature somehow extractable (i.e., we could extract the secret from the view of any signing algorithm). Second, sending a bitstring as a challenge must be done "at once" as we cannot afford loosing time by sending bits sequentially. This requires to send all bits in parallel. Finally, the signature operation would take too much time on any commercial device to make the protocol work with a B low enough. Indeed, the crucial problem in making a PoPoK scheme is to remove all computation from the time-critical challenge-response exchange and to split it into boolean challenge-response rounds.

Example 9. We recall the Brands-Chaum protocol [12] on Fig. 4. It is based on a commitment scheme and a signature scheme. The Brands-Chaum protocol is not sound if the signature resists to chosen-message attacks: a far-away malicious prover can let a close-by actor run the protocol then sign the transcript for him.

Example 10. We recall the Hermans-Peeters-Onete protocol [23] on Fig. 5. It is based on the one-more discrete logarithm problem and the decisional Diffie-Hellman problem over elliptic curves. We can easily see that it is not sound. Indeed, the adversary can just relay messages between the prover and the verifier and run the challenge phase after being given the vectors a'_0 and a'_1. These vectors do not leak any sensitive information.

Example 11. We can easily construct a PoPoK based on a symmetric distance bounding protocol by making the shared secret used in distance-bounding be the result on a public-key based key agreement [11,29]. However, this construction does not provide soundness since leaking the shared secret does not imply leaking the secret key.

4 ProProx: a PoPoK Scheme

4.1 Building Blocks

Perfectly binding bit commitment. Depending on the security parameter λ, we use a (multiplicative) group structure with two Abelian groups L and G and an element θ such that G is generated by L and θ, $\theta \notin L$, and L is the set of all squares of G. We further assume that it is easy to do group operations and comparisons in G and to sample elements in G uniformly.[6] Finally, we assume it is computationally hard to distinguish elements from L and from G.

We define $\mathsf{Com}(b;\rho) = \theta^b\rho^2$ for a bit b and a random $\rho \in G$, like in the Goldwasser-Micali cryptosystem [21,22]. So, Com is computationally hiding as defined by Def. 19. We will not require any secret key to extract b, although there *exists* a function Com^{-1} such that $\mathsf{Com}^{-1}(\mathsf{Com}(b;\rho)) = b$ for all $b \in \{0,1\}$ and $\rho \in G$. We will rather use the homomorphic properties of the commitment and prove the correct commitment in a zero-knowledge way. Def. 19 in appendix formally defines Com.

For instance, we can take a Blum integer N, i.e., $N = PQ$ for two distinct primes P and Q which are congruent to 3 modulo 4. We set L to the set of quadratic residues modulo N and θ a residue modulo N such that $\left(\frac{\theta}{P}\right) = \left(\frac{\theta}{Q}\right) = -1$. E.g., $\theta = -1$. The algorithm Com is given N and θ. We sample $r \in G$ by $r = \theta^b\rho^2 \bmod N$, for $b \in \mathbf{Z}_2$ and $\rho \in \mathbf{Z}_N^*$. Distinguishing G from L is the quadratic residuosity problem, which is supposed to be hard. In this case, N is assumed to come from a Common Reference String (CRS).

A zero-knowledge proof for z being a square. We use the Fiat-Shamir protocol [18]. Namely, we show that z is a commitment to zero with a witness ζ (i.e., $z = \zeta^2$) with the protocol from Fig. 3, based on a perfectly hiding trapdoor commitment. Concretely, we use Def. 20 and Def. 21 in appendix with an $\mathcal{NP}$ language L. This is the set of all squares. If $z = \zeta^2$, we say that z is a member of L with witness ζ.

The protocol of Fig. 3 in appendix is $\frac{1}{2}$-sound and zero-knowledge. It must be run k times in parallel to achieve a soundness level $\kappa = 2^{-k}$. We denote it by $\mathsf{ZKP}_\kappa(z : \zeta)$.

By using parallel composition, we extend the protocol to prove that $z_1, \ldots, z_k$ are some commitments to zero with witness $\zeta_1, \ldots, \zeta_k$ respectively, and denote it by $\mathsf{ZKP}_\kappa(z_1, \ldots, z_k : \zeta_1, \ldots, \zeta_k)$. I.e., it succeeds with probability up to κ if there exists i such that $z_i \notin L$.

(Perfectly binding) deterministic commitment. Given a hash function H making coins for Com, we define a deterministic commitment by $\mathsf{Com}_H(x) = (\mathsf{Com}(x_1;H(x,1)), \ldots, \mathsf{Com}(x_s;H(x,s)))$ for $x \in \mathbf{Z}_2^s$. We assume that Com_H is a one-way function. Def. 22 in appendix formally defines Com_H.

When H is a random oracle, we can easily show that Com_H satisfies Def. 22. In other cases, we must assume that Com composed with H is one-way. Constructions without using a random oracle are left to future work.

4.2 The ProProx Protocol

We define the ProProx protocol, as depicted on Fig. 1. We consider s (the size of the secret), n (the number of rounds per iteration), τ (the minimal number of correct rounds per iteration for acceptance) as functions in terms of the security parameter λ. We assume s and n are asymptotically linear. We also use a vector $b \in \mathbf{Z}_2^s$. We consider the vector b as fixed in the protocol. For now on, b is not important. It will only appear in Lemma 17 to treat distance fraud. There, we will only require b to have a Hamming weight of $\left\lfloor \frac{n}{2} \right\rfloor$.

[6] So, we can sample an element of L uniformly by taking r^2 with r uniformly selected in G.

Verifier		**Prover**
public: y	$y = \mathsf{Com}_H(x)$	secret: x
	initialization phase for $j = 1$ to s in parallel	
		pick $a_{i,j} \in \mathbf{Z}_2$, $\rho_{i,j}$, $i = 1, \ldots, n$
	$\xleftarrow{A_{1,j},\ldots,A_{n,j}}$	$A_{i,j} = \mathsf{Com}(a_{i,j}; \rho_{i,j})$
	challenge phase for $i = 1$ to n and $j = 1$ to s	
pick $c_{i,j} \in \mathbf{Z}_2$		
start $\mathsf{timer}_{i,j}$	$\xRightarrow{c_{i,j}}$	receive $c'_{i,j}$
receive $r_{i,j}$, stop $\mathsf{timer}_{i,j}$	$\xLeftarrow{r'_{i,j}}$	$r'_{i,j} = a_{i,j} + c'_{i,j} b_i + c'_{i,j} x_j$
	verification phase	
check $\#I_j = \tau$, $\mathsf{timer}_{i,j} \leq 2B$ for $i \in I_j$, $j = 1, \ldots, s$	$\xleftrightarrow{\text{agree on } I_1,\ldots,I_s}$	
$z_{i,j} = A_{i,j} \left(\theta^{b_i} y_j\right)^{c_{i,j}} \theta^{-r_{i,j}}$	$\xleftrightarrow{ZKP_\kappa(z_{i,j}:\zeta_{i,j}:i \in I_j, j=1,\ldots,s)}$	$\zeta_{i,j} = \rho_{i,j} H(x,j)^{c'_{i,j}}$
	$\xrightarrow{\mathsf{Out}_V}$	

Fig. 1. ProProx: a Sound and Secure PoPoK.

The prover holds a secret $x \in \mathbf{Z}_2^s$ and the public key is $y = \mathsf{Com}_H(x)$. We iterate s times and in parallel a protocol which we call an iteration and which corresponds to an index j. First, the prover selects n bits $a_{1,j}, \ldots, a_{n,j} \in \mathbf{Z}_2$ and commits to them using some fresh coins $\rho_{1,j}, \ldots, \rho_{n,j}$, respectively. So, $A_{i,j} = \mathsf{Com}(a_{i,j}; \rho_{i,j})$, $i = 1, \ldots, n$. The $A_{i,j}$'s are sent to the verifier.

In the challenge phase, we have n time-critical rounds (in each iteration). These rounds may be subject to noise. For $i = 1, \ldots, n$, the verifier picks a challenge $c_{i,j} \in \mathbf{Z}_2$ at random and sends it to the prover. The prover receives $c'_{i,j}$ (which may be different, due to noise). He computes his response $r'_{i,j} = a_{i,j} + c'_{i,j} b_i + c'_{i,j} x_j$ and sends it back to the verifier at once. The verifier receives $r_{i,j}$. The verifier measures the elapsed time $\mathsf{timer}_{i,j}$ taken to receive $r_{i,j}$ after $c_{i,j}$ was sent. Below, p_{noise} is the probability that some noise corrupts a challenge/response round. We assume that the noise corrupts each round independently.

The $c'_{i,j} \mapsto r'_{i,j}$ function maps one bit to one bit. Its table could be precomputed to save time. Furthermore, these $c'_{i,j} \mapsto r'_{i,j}$ rounds could be run by analog circuits to avoid communication delays.

In the verification phase, the prover and the verifier determine a set I_j of τ round indices which they believe are correct. The way this agreement is done is not important (as long as the prover does not leak). Then, the verifier checks whether I_j has cardinality τ and the corresponding timers are small enough. If this fails, the verifier rejects. As a concrete instance for I_j agreement, we suggest that the prover sends (through the lazy noiseless channel) the $c'_{i,j}$ and $r'_{i,j}$ to the verifier. The verifier then takes the first τ rounds for which $c_{i,j} = c'_{i,j}$, $r_{i,j} = r'_{i,j}$, and $\mathsf{timer}_{i,j} \leq 2B$ to define I_j and sends I_j to the prover. If there are not enough correct rounds, the protocol aborts.

Next, the prover and the verifier run the interactive proof ZKP_κ to show that the responses $r_{i,j}$'s are consistent with the $A_{i,j}$'s and y_j's. Namely, for all j and $i \in I_j$, they compute

$$z_{i,j} = A_{i,j} \left(\theta^{b_i} y_j\right)^{c_{i,j}} \theta^{-r_{i,j}} \quad , \quad \zeta_{i,j} = \rho_{i,j} H(x,j)^{c'_{i,j}}$$

Since $A_{i,j} = \theta^{a_{i,j}} \rho_{i,j}^2$ and $y_j = \theta^{x_j} H(x,j)^2$, it is easy to verify that $r_{i,j} = a_{i,j} + c_{i,j} b_i + c_{i,j} x_j$ is equivalent to the existence of $\zeta_{i,j}$ such that $z_{i,j} = \zeta_{i,j}^2$. That is, $z_{i,j} \in L$. If this fails, the protocol aborts. When the protocol aborts, the verifier sends $\mathsf{Out}_V = 0$. Otherwise, he sends $\mathsf{Out}_V = 1$.

4.3 Analysis

Theorem 12. *Let $\varepsilon > 0$ be a constant. We assume that n is linear in λ and that $\frac{\tau}{n} < 1 - p_{\mathsf{noise}} - \varepsilon$ or that $p_{\mathsf{noise}} = 0$. Under the assumption that Com is a homomorphic bit commitment [Def. 19] and that ZKP_κ is complete [Def. 20], the ProProx protocol is a PoPoK when the challenge/response rounds are subject to a noise level of p_{noise} [Def. 2].*

We further assume that $\tau \geq n - (\frac{1}{2} - 2\varepsilon)\lceil \frac{n}{2} \rceil$. Under the assumption that Com is a perfectly binding, computationally hiding, and homomorphic bit commitment [Def. 19], that Com_H is one-way [Def. 22], and that ZKP_κ is a κ-sound [Def. 20] computationally zero-knowledge [Def. 21] proof of membership for $\kappa = \mathsf{negl}(\lambda)$, the ProProx protocol is a sound [Def. 5] and secure [Def. 4] PoPoK.

Proof. Completeness for $p_{\mathsf{noise}} = 0$ is trivial. Proving completeness when $^{\tau}$

Thanks to Lemma 1, when $r_{i,j}$ comes from a close-by participant, we can write

$$r_{i,j} = \mathsf{Algo}(\mathsf{View}_{i,j}, c_{i,j}, \mathsf{Other}_{i,j})$$

with $\mathsf{View}_{i,j}$ the partial view (before being able to see $c_{i,j}$) of close-by participants and messages $\mathsf{Other}_{i,j}$ independent (in the sense of Lemma 1) from $c_{i,j}$ coming to these participants from far-away. Note that both $\mathsf{View}_{i,j}$ and $\mathsf{Other}_{i,j}$ can be computed from the final views of the close-by participants. So, thanks to Lemma 1, we can compute in this case both $\mathsf{resp}(0) = \mathsf{Algo}(\mathsf{View}_{i,j}, 0, \mathsf{Other}_{i,j})$ and $\mathsf{resp}(1) = \mathsf{Algo}(\mathsf{View}_{i,j}, 1, \mathsf{Other}_{i,j})$ without rewinding (i.e., from the final view only). So, we can compute the guess $\xi_{i,j} = \mathsf{resp}(1) - \mathsf{resp}(0) - b_i$ for x_j.

Note that if the answer $r_{i,j}$ comes to $\mathcal{V}$ from far-away, we can still apply Lemma 1 and deduce that the answer is the same for $c_{i,j} = 0$ and $c_{i,j} = 1$: $\mathsf{resp}(0) = \mathsf{resp}(1)$. Although we may be unable to compute the responses, we can still compute $\xi_{i,j} = -b_i$. In all cases, we can always compute the vectors $\xi_j = (\xi_{1,j}, \ldots, \xi_{n,j})$.

The extractor is taking Algo to compute ξ_j and to deduce $x'_j = \mathsf{majority}(\xi_j)$.

Given c, if $a_{i,j} + cb_i + cx_j = \mathsf{resp}(c)$, we say that the answer to $c_{i,j} = c$ is correct. Let $R_j = (R_{1,j}, \ldots, R_{n,j})$ where $R_{i,j}$ is the number of challenge values $c \in \{0,1\}$ for which the answer is correct for $c_{i,j} = c$. If $R_{i,j} = 2$, i.e. if we have two correct answers for $c_{i,j} = 0$ and for $c_{i,j} = 1$, then we have $\xi_{i,j} = x_j$. If $R_{i,j} = 1$, we have $\xi_{i,j} \neq x_j$. Let $\mathcal{R}$ be the set of all $R = (R_1, \ldots, R_n) \in \{0,1,2\}^n$ such that the number of i's with $R_i = 2$ is at least $\lfloor \frac{n}{2} \rfloor + 1$. For $R_j \in \mathcal{R}$, we have a majority of i's for which $\xi_{i,j} = x_j$. So, we have $x'_j = x_j$.

We let $R = (R_1, \ldots, R_s)$ be a random variable defining the R_j vectors. Since $\mathcal{V}$ is selecting the challenges at random, once $R_{i,j}$ is determined, we can compute the probability that the answer to $c_{i,j}$ is correct: if $R_{i,j} = 2$ then this probability is 1. Otherwise, it is at most $\frac{1}{2}$. If W is the random variable giving the number of j such that $R_j \notin \mathcal{R}$, we have $\Pr[S|W = w] \leq p_B^w$. So, $\Pr[W = w, S] \leq p_B^w \Pr[W = w]$ and then $\Pr[W \geq w, S] \leq p_B^w$. Finally, by splitting with the S and $\neg S$ events, we have

$$\Pr[W \geq w, \mathcal{A} \text{ accepts}] \leq \Pr[\neg S, \mathcal{A} \text{ accepts}] + \Pr[W \geq w, S] \leq \mu + p_B^w$$

We note that each index j such that $x'_j \neq x_j$ must corresponds to $R_j \notin \mathcal{R}$. So, having that $\mathcal{A}$ accepts and the extractor gives at least w errors occurs with probability bounded by $\mu + p_B^w$. □

Lemma 14 (Soundness). *We assume that n is linear in λ and that $\tau \geq n - (\frac{1}{2} - 2\varepsilon)\lceil \frac{n}{2} \rceil$ for some constant ε and some parameter $\lceil \frac{n}{2} \rceil$. Under the assumption that Com is a perfectly binding homomorphic bit commitment, and that ZKP_κ is a κ-sound proof of membership for $\kappa = \mathsf{negl}(\lambda)$, the ProProx protocol is a sound proof of proximity.*

Proof. We define $p_B = \mathsf{Tail}(\lceil \frac{n}{2} \rceil, \tau - \lfloor \frac{n}{2} \rfloor, \frac{1}{2})$. We assume $p \geq p_{\mathsf{Sound}}$ where

$$p_{\mathsf{Sound}} = 2(p_B^w + \kappa) \tag{2}$$

for $w > 0$ constant. Actually, to use (2) with concrete parameters, w is chosen as the maximal value such that we want to protect against an adversary who can afford an exhaustive search of s^w steps. Since $\frac{\tau - n + \lceil \frac{n}{2} \rceil}{\lceil \frac{n}{2} \rceil} \geq \frac{1}{2} + 2\varepsilon$, we have $p_B \leq e^{-8\varepsilon^2 n}$ due to the Chernoff-Hoeffding bound (see Appendix A), which is negligible. So, p_{Sound} is negligible.

We can use the extractor of Lemma 13 on views taken from an experiment run. If $\mathcal{V}$ rejects, the extraction produces nothing. We iterate this extraction $O(\frac{1}{p})$ times until one experiment succeeds. So, we obtain for sure a guess x' for x (with possible errors). The probability that at least w errors occurs

in the extracted pairs is bounded by $\frac{p_B^w+\kappa}{p} \leq \frac{1}{2}$. When there are less errors, we can correct them by exhaustive search in time $O(s^w)$ (which is polynomial). If this fails (i.e., if it gives no preimage of y by Com_H), as some extracted pairs may have too many errors, we can just iterate. With a constant number of iterations, we finally extract x. The overall expected complexity is $\mathsf{Poly}(\lambda)/p$. □

Our technique to prove security relies on Lemma 13 and zero-knowledge. We state this latter property.

Lemma 15 (Zero-knowledge). *Under the assumption that* Com *is a computationally hiding bit commitment and that* ZKP_κ *is a computationally zero-knowledge proof of membership, The ProProx protocol is zero-knowledge following Def. 21.*

Proof. We have to prove that, given two participants $P(x)$ and $V^*(y,\mathsf{aux})$, there exists a simulator $S(y,\mathsf{aux})$ such that $V^*(y,\mathsf{aux}) \leftrightarrow P(x)$ produces a view of $V^*(y,\mathsf{aux})$ which is computationally indistinguishable from the output of $S(y,\mathsf{aux})$. We will actually construct a sequence of simulations. We define an interactive $V'(y,\mathsf{aux})$ to replace $V^*(y,\mathsf{aux})$, and some interactive $P'(x)$ and P'' to replace $P(x)$.

We denote $\bar{z}$ the vector of all $z_{i,j}$ for $j = 1,\ldots,s$ and $i \in I_j$, and $\bar{\zeta}$ the vector of all $\zeta_{i,j}$. We split $V^*(y,\mathsf{aux})$ into two protocols $V_1(y,\mathsf{aux})$ and $V_2(\bar{z},\mathsf{aux}')$, where V_1 mimics V^* until the $\mathsf{ZKP}_\kappa(\bar{z}:\bar{\zeta})$ protocol must start. Actually, V_2 executes only $\mathsf{ZKP}_\kappa(\bar{z}:\bar{\zeta})$ where aux' is the final view of $V_1(y,\mathsf{aux})$. The final view of $V_2(\bar{z},\mathsf{aux}')$ is of form $v = (\bar{z},\mathsf{aux}',t)$. We write $g(v) = (\mathsf{aux}',t)$, which is the final view of $V^*(y,\mathsf{aux})$. Similarly, we split $P(x)$ into $P_1(x)$ and $P_2(x,u)$ where (x,u) is the view of $P_1(x)$. Clearly, running either $V^*(y,\mathsf{aux}) \leftrightarrow P(x)$ and taking the final view of V^*, or $V_1(y,\mathsf{aux}) \leftrightarrow P_1(x)$, $V_2(\bar{z},\mathsf{aux}') \leftrightarrow P_2(x,u)$, then taking $g(v)$ is the same. This simulation is illustrated on the left-hand side of Fig. 2.

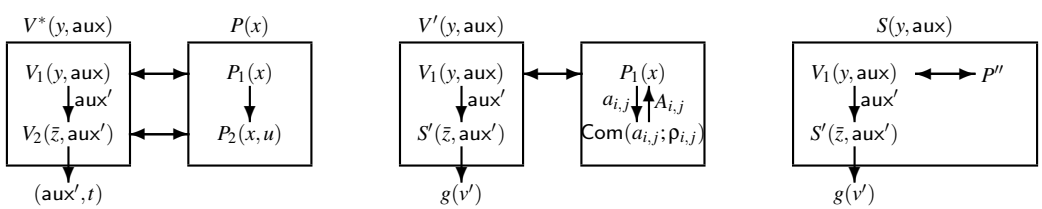

Fig. 2. Applying a ZK Reduction.

First, $V'(y,\mathsf{aux})$ runs a simulation of $V_1(y,\mathsf{aux})$ interacting with $P_1(x)$. Then, $V'(y,\mathsf{aux})$ runs the simulator $S'(\bar{z},\mathsf{aux}')$ of the $\mathsf{ZKP}_\kappa(\bar{z}:\bar{\zeta})$ protocol associated to the verifier $V_2(\bar{z},\mathsf{aux}')$. Let v' be the output of $S'(\bar{z},\mathsf{aux}')$. Finally, $V'(y,\mathsf{aux})$ produces $g(v')$ as an output. This simulation is illustrated on the middle of Fig. 2. Due to the zero-knowledge property of $\mathsf{ZKP}_\kappa(\bar{z}:\bar{\zeta})$, v' is computationally indistinguishable from the final view of $V_2(\bar{z},\mathsf{aux}')$. So, the final view of $V'(y,\mathsf{aux})$ in $V'(y,\mathsf{aux}) \leftrightarrow P_1(x)$ and the final view of $V^*(y,\mathsf{aux})$ in $V^*(y,\mathsf{aux}) \leftrightarrow P(x)$ are indistinguishable.

Note that $P_1(x)$ makes no longer extra use of the coins ρ_i's (as $P_2(x,u)$ does in ZKP_κ). So, the commitment can be outsourced to a challenger playing the real-or-random hiding game for Com. We modify $P_1(x)$ into an algorithm $P'(x)$ who sets $A_{i,j}$ to the commitment to some random bit instead of $a_{i,j}$. Thanks to the hiding property of Com, the output of $V'(y,\mathsf{aux}) \leftrightarrow P_1(x)$ and of $V'(y,\mathsf{aux}) \leftrightarrow P'(x)$ are indistinguishable.

Finally, r_i' in $P'(x)$ is now uniformly distributed and independent from all the rest, so we change $P'(x)$ into an algorithm P'' which sends a random r_i' instead. Note that P'' no longer needs x. So, the

view of V^* in $V^*(y, \mathsf{aux}) \leftrightarrow P(x)$ and the output of $V'(y, \mathsf{aux}) \leftrightarrow P''$ are indistinguishable. This defines a simulator $S(y, \mathsf{aux})$, as illustrated on the right-hand-side of Fig. 2.

Overall, the advantage to distinguish is bounded by

$$p_{\mathsf{ZK}} = p_{\mathsf{ZKP}} + ns.p_{\mathsf{Com}} \tag{3}$$

where p_{ZKP} is the advantage to distinguish the ZK simulation from real in the ZKP protocol and p_{Com} is the bound on the hiding property of Com. □

Lemma 16 (Security). *We assume that s is linear and that $\tau \geq n - (\frac{1}{2} - 2\varepsilon)\lceil \frac{n}{2} \rceil$ for some constant ε. Under the assumption that* Com *is a perfectly binding, and computationally hiding homomorphic bit commitment, that Com_H is one-way, and that ZKP_κ is a κ-sound computationally zero-knowledge proof of membership for $\kappa = \mathsf{negl}(\lambda)$, the ProProx protocol is secure following Def. 4.*

Proof. We consider an experiment exp with an honest always far-away prover. Let p be the probability that $\mathcal{V}$ accepts. We want to show that $p = \mathsf{negl}(\lambda)$.

We define $p_B = \mathsf{Tail}(\lceil \frac{n}{2} \rceil, \tau - \lfloor \frac{n}{2} \rfloor, \frac{1}{2})$ and $w > 0$ constant. We use the extractor of Lemma 13 to extract the vector x' when $\mathcal{V}$ accepts, with at least w errors to x with probability bounded by $p_B^w + \kappa$. Then, by a $O(s^w)$-time exhaustive search on the errors, we correct x' and check if we obtain a preimage of Com_H. This gives x in polynomial time and a probability of success of at least $p - p_B^w - \kappa$, by playing with some non-concurrent instances of $P(x)$. For each of the non-concurrent instances of $P(x)$, we use the zero-knowledge property of $P(x)$ to construct an algorithm inverting Com_H with probability of success of at least $p - p_B^w - \kappa - r.p_{\mathsf{ZK}}$, where r is the number of $P(x)$ instances in one experiment. By assumption on Com_H, this must be bounded by some negligible p_{Com}. So, we have $p \leq p_{\mathsf{Sec}}$ with

$$p_{\mathsf{Sec}} = p_B^w + \kappa + r \cdot p_{\mathsf{ZK}} + p_{\mathsf{Com}} \tag{4}$$

Since $\frac{\tau - n + \lceil \frac{n}{2} \rceil}{\lceil \frac{n}{2} \rceil} \geq \frac{1}{2} + 2\varepsilon$, we have $p_B \leq e^{-8\varepsilon^2 n}$ due to the Chernoff-Hoeffding bound (see Appendix A), which is negligible. The values κ, p_{ZK}, and p_{Com} are also negligible, while r is polynomial and w is constant. So, p_{Sec} is negligible. □

Note that a malicious prover can run a distance fraud in each round where $b_i = x_j$, as $r_{i,j}$ no longer depends on $c_{i,j}$. For $x = 0$ (as allowed in the malicious prover model) and $b = 0$, this can be done in all rounds, so we can have a distance fraud. There is no contradiction with soundness: an observer seeing the verifier accepts can deduce that x_j is likely to be zero, for all j. So, the malicious prover leaks.

To have distance fraud resistance, we adopt the following trick taken from DB2 [9,10]: we select a vector b with Hamming weight $\lfloor \frac{n}{2} \rfloor$ so that half of the rounds will really use $c_{i,j}$. Actually, b has a maximal distance to the repetition code.

Lemma 17 (DF-Resistance). *We assume that n is linear and that $\tau \geq n - (\frac{1}{2} - 2\varepsilon)\lfloor \frac{n}{2} \rfloor$ for some constant ε and that the vector b has a Hamming weight of $\lfloor \frac{n}{2} \rfloor$. Under the assumption that* Com *is a perfectly binding bit commitment and that ZKP_κ is a κ-sound computationally zero-knowledge proof of membership for $\kappa = \mathsf{negl}(\lambda)$, the ProProx protocol resists to distance fraud following Def. 6.*

Proof. Due to the perfectly binding property, the view of $\mathcal{V}$ uniquely defines x_j and $a_{i,j}$. Thanks to Lemma 1, $r_{i,j}$ is independent (in the sense of Lemma 1) from $c_{i,j}$. So, for $b_i \neq x_j$ (which happens for half of the rounds), we have that $\Pr[r_{i,j} = a_{i,j} + c_{i,j} b_i + c_{i,j} x_j] = \frac{1}{2}$. So, the probability that the statement in ZKP_κ holds is bounded by $\mathsf{Tail}(\lfloor \frac{n}{2} \rfloor, \tau - \lceil \frac{n}{2} \rceil, \frac{1}{2})^s$ which is negligible for $\frac{\tau - n + \lfloor \frac{n}{2} \rfloor}{\lfloor \frac{n}{2} \rfloor} \geq \frac{1}{2} + 2\varepsilon$, due to the Chernoff-Hoeffding bound (see Appendix A). Due to the fact that ZKP_κ is sound, the verifier accepts

with probability bounded by $\kappa + \mathsf{Tail}(\lfloor \frac{n}{2} \rfloor, \tau - \lceil \frac{n}{2} \rceil, \frac{1}{2})^s$. Finally, the attack succeeds with probability bounded by

$$p_{\mathsf{DF}} = \kappa + \mathsf{Tail}\left(\left\lfloor \frac{n}{2} \right\rfloor, \tau - \left\lceil \frac{n}{2} \right\rceil, \frac{1}{2}\right)^s \tag{5}$$

which is also negligible. □

We could also treat distance hijacking [15] specifically. As we can see in our protocol, the prover responds using his own random $a_{i,j}$ which are hidden in the commitment. So, is an honest prover with another identity interact with $\mathcal{V}$ during the challenge phase, his response would match the one that $\mathcal{V}$ expect from his view, and the soundness of ZKP would make the protocol fail. The full formalization is left to future work.

4.4 A Variant for Noiseless Communications

The protocol could be simplified in noiseless environment. For this, we would take $n = \tau$. There is clearly no need to agree on I_j which is always the full set $I_j = \{1, \ldots, n\}$. The protocol is much simpler.

4.5 Concrete Parameters

To see if the proven bounds Eq. (2), Eq. (4), and Eq. (5) may tight or not, we look at the best known attacks. They correspond to the following probabilities of success:

$$p_{\mathsf{DF}} = \mathsf{Tail}\left(\left\lfloor \frac{n}{2} \right\rfloor, \tau - \left\lceil \frac{n}{2} \right\rceil, \frac{1}{2}\right)^s \quad , \quad p_{\mathsf{Sec}} = \mathsf{Tail}\left(n, \tau, \frac{1}{2}\right)^s \quad , \quad p_{\mathsf{Sound}} = \mathsf{Tail}\left(\left\lceil \frac{n}{2} \right\rceil, \tau - \left\lfloor \frac{n}{2} \right\rfloor, \frac{1}{2}\right)^s$$

The DF attack with success probability p_{DF} consists of guessing c_i in half of the rounds for which $b_i \neq x_j$. So, the proven bound Eq. (5) is pretty tight.

The MF attack with success probability p_{Sec} follows the post-ask strategy: the adversary first guesses the answers to all challenges then play with the prover with the same challenges. Clearly, there is a gap between p_{Sec} and the proven bound of Eq. (4).

The TF attack with success probability p_{TF} consists of giving a table of all $c'_{i,j} \mapsto r'_{i,j}$ which is corrupted in half of the rounds in each iteration, so that it gives no information about x_j. There is also a gap with the proven bound Eq. (2).

So, it may be the case that either the bounds Eq. (4) and Eq. (2) can be improved, or that there exist better attacks. To select the parameters, we could either use the *proven* bounds or the equations based on the best known attacks that we call the *empirical* bounds.

As concrete parameters, we could suggest $s = 80$ bits as the size of the secret and a modulus N of 2048 bits. Then, we look for n and τ which minimize the total number of rounds n while keeping $p_{\mathsf{Comp}} \approx 1 - 2^{-7}$ and different objectives: we propose several vectors of parameters to reach the online security of either $\sigma = 2^{-20}$ (*high*) or $\sigma = 2^{-10}$ (*low*), with *proven* bounds or *empirical* bound, and with either $p_{\mathsf{noise}} = 1\%$ or the noiseless variant ($p_{\mathsf{noise}} = 0$) from Section 4.4. In the computation of Eq. (2) and Eq. (4), we took $\kappa = \frac{\sigma}{4}$ and w such that the exhaustive search is not more for a random s-bit string, i.e., $s^w \leq 2^s$. So, we took $w = \lfloor \frac{s \log 2}{\log s} \rfloor$.

The total number of rounds is ns.

security	bounds	p_{noise}	ns	s	n	w	τ	p_{Comp}	p_{DF}	p_{Sec}	p_{Sound}
high	proven	1%	1 360	80	17	12	14	$1-2^{-9}$	2^{-22}	2^{-22}	2^{-21}
high	empirical	1%	720	80	9	–	7	$1-2^{-7}$	2^{-43}	2^{-278}	2^{-80}
low	proven	1%	720	80	9	12	7	$1-2^{-7}$	2^{-12}	2^{-11}	2^{-10}
low	empirical	1%	720	80	9	–	7	$1-2^{-7}$	2^{-43}	2^{-278}	2^{-80}
high	proven	0	240	80	3	12	3	1	2^{-22}	2^{-22}	2^{-21}
high	empirical	0	160	80	2	–	2	1	2^{-80}	2^{-160}	2^{-80}
low	proven	0	160	80	2	12	2	1	2^{-12}	2^{-11}	2^{-10}
low	empirical	0	160	80	2	–	2	1	2^{-80}	2^{-160}	2^{-80}
low	proven	0	80	80	1	12	1	1	$(2^{-22})^*$	2^{-12}	2^{-11}
high	empirical	0	80	80	1	–	1	1	$(2^{-22})^*$	2^{-80}	2^{-80}

Clearly, there is a big gap between proven and empirical parameters in the high security values. In the noiseless variant, the round complexity looks large but acceptable. ProProx may be hard to implement on NFC credit cards, but, at least all vectors in the noiseless variant and all vectors with the empirical bounds are feasible for implementation on an NFC smartphone for payment applications.

In theory, we cannot consider n as a constant since the bounds in Eq. (2) and Eq. (4) are not negligible in this case. However, it makes sense to consider very low n in practice if we estimate the exact security.

As we can see, n is very low in the noiseless variant. Typically, $n = 2$. We cannot have $n = 1$ due to the p_{DF} bound: with $n = 1$, b has a single bit and for $x_j = b$ for all j, the distance fraud is possible. However, we could tweak the protocol to obtain DF-resistance for $n = 1$: we could have b selected by the verifier for each iteration at random. I.e., V selects a random vector $b \in \mathbf{Z}_2^s$ and sends it to P during the initialization. Then, P uses $r_{i,j} = a_{i,j} + c_{i,j}(b_j + x_j)$ as a response function. Clearly, our results about security and soundness do not depend on the selection of b, so they remain valid with this variant. With this variant, we could prove $p_{\mathsf{DF}} = \kappa + (\frac{3}{4})^s \approx 2^{-22}$ by standard techniques. This selection is indicated under parenthesis in the table as it requires a change in the protocol and does not use Eq. (5).

As we can see, the noiseless case with $n = 1$ and $s = 80$ offers pretty reasonable parameters. We can prove the *low* security level and we can even hope that it provides a *high* one, due to the remaining gap between *proven* and *empirical* bounds. An interesting open question is to wonder if we could have protocols with less than s rounds. In our construction, we build an extractor using a single session of the protocol by $\mathcal{V}$. We conjecture and in such case, to extract an s-bit secret, we need to get s bits in the challenge phase. So our protocol could be optimal among those with a single-session extractor.

5 Conclusion

We proposed ProProx, the very first PoPoK addressing soundness. It is provably secure. Conceptually, we believe it could integrate well in an infrastructure for contactless payments. A remaining challenge is to construct a more efficient PoPoK. Another open question would be to have a tight security proof for ProProx.

References

1. G. Avoine, M. Bingöl, S. Kardas, C. Lauradoux, B. Martin. A Framework for Analyzing RFID Distance Bounding Protocols. *Journal of Computer Security*, vol. 19(2), pp. 289–317, 2011.
2. G. Avoine, A. Tchamkerten. An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement. In *Information Security ISC'09*, Pisa, Italy, Lecture Notes in Computer Science 5735, pp. 250–261, Springer-Verlag, 2009.

3. A. Bay, I. Boureanu, A. Mitrokotsa, I. Spulber, S. Vaudenay. The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In *INSCRYPT'12*, Beijing, China, Lecture Notes in Computer Science 7763, pp. 371–391, Springer-Verlag, 2012.
4. T. Beth, Y. Desmedt. Identification Tokens or: Solving The Chess Grandmaster Problem. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 537, pp. 169–176, Springer-Verlag, 1991.
5. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Secure & Lightweight Distance-Bounding. In *Lightweight Cryptography for Security and Privacy LightSec'13*, Gebze, Turkey, Lecture Notes in Computer Science 8162, pp. 97–113, Springer-Verlag, 2013.
6. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Practical & Provably Secure Distance-Bounding. To appear in the Journal of Computer Security. Available as Eprint technical report, 2013. http://eprint.iacr.org/2013/465.pdf
7. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Towards Secure Distance Bounding. In *Fast Software Encryption'13*, Singapore, Lecture Notes in Computer Science 8424, pp. 55–67, Springer-Verlag, 2013.
8. I. Boureanu, A. Mitrokotsa, S. Vaudenay. Practical & Provably Secure Distance-Bounding. To appear in the proceedings of ISC'13.
9. I. Boureanu, S. Vaudenay. Optimal Proximity Proofs. Eprint technical report, 2014. http://eprint.iacr.org/2014/693.pdf
10. I. Boureanu, S. Vaudenay. Optimal Proximity Proofs. To appear in the Proceedings of Inscrypt'14.
11. I. Boureanu, S. Vaudenay. Challenges in Distance-Bounding. To appear in the IEEE Security & Privacy Magazine, 2015.
12. S. Brands, D. Chaum. Distance-Bounding Protocols (Extended Abstract). In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norway, Lecture Notes in Computer Science 765, pp. 344–359, Springer-Verlag, 1994.
13. L. Bussard, W. Bagga. Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks. In *IFIP TC11 International Conference on Information Security SEC'05*, Chiba, Japan, pp. 223–238, Springer, 2005.
14. H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Annals of Mathematical Statistics*, vol. 23 (4), pp. 493-507, 1952.
15. C.J. F. Cremers, K.B. Rasmussen, B. Schmidt, S. Čapkun. Distance Hijacking Attacks on Distance Bounding Protocols. In *IEEE Symposium on Security and Privacy S&P'12*, San Francisco, California, USA, pp. 113–127, IEEE Computer Society, 2012.
16. Y. Desmedt. Major Security Problems with the "Unforgeable" (Feige-)Fiat-Shamir Proofs of Identity and How to Overcome Them. In *Congress on Computer and Communication Security and Protection Securicom'88*, Paris, France, pp. 147–159, SEDEP Paris France, 1988.
17. U. Dürholz, M. Fischlin, M. Kasper, C. Onete. A Formal Approach to Distance-Bounding RFID Protocols. In *Information Security ISC'11*, Xi'an, China, Lecture Notes in Computer Science 7001, pp. 47–62, Springer-Verlag, 2011.
18. A. Fiat, A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology CRYPTO'86*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 263, pp. 186–194, Springer-Verlag, 1987.
19. M. Fischlin, C. Onete. Terrorism in Distance Bounding: Modelling Terrorist-Fraud Resistance. In *Applied Cryptography and Network Security ACNS'13*, Banff AB, Canada, Lecture Notes in Computer Science 7954, pp. 414–431, Springer-Verlag, 2013.
20. S. Gambs, C. Onete, J.-M. Robert. Prover Anonymous and Deniable Distance-Bounding Authentication. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'14)*, Kyoto, Japan, pp. 501–506, ACM Press, 2014.
21. S. Goldwasser, S. Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, San Fransisco, California, U.S.A., pp. 365–377, ACM Press, 1982.
22. S. Goldwasser, S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, vol. 28, pp. 270–299, 1984.
23. J. Hermans, R. Peeters, C. Onete. Efficient, Secure, Private Distance Bounding without Key Updates. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks WISEC'13*, Budapest, Hungary, pp. 195–206, ACM, 2013.
24. W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, vol. 58, pp. 13–30, 1963.
25. A. Ranganathan, N.O. Tippenhauer, B Škorić, D. Singelée, S. Čapkun. Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System. In *Computer Security - ESORICS'12*, Pisa, Italy, Lecture Notes in Computer Science 7459, pp. 415–432, Springer-Verlag, 2012.
26. A. Ranganathan, B. Danev, S. Čapkun. Low-Power Distance Bounding. Available as CoRR 1404.4435 technical report, Cornell University 2014. http://arxiv.org/abs/1404.4435

27. K.B. Rasmussen, S. Čapkun. Realization of RF Distance Bounding. In *USENIX Security Symposium (USENIX'10)*, Washington, DC, USA, pp. 389–402, USENIX, 2010.
28. S. Vaudenay. On Modeling Terrorist Frauds. In *Provable Security ProvSec'13*, Melaka, Malaysia, Lecture Notes in Computer Science 8209, pp. 1–20, Springer-Verlag, 2013.
29. S. Vaudenay. Private and Secure Public-Key Distance Bounding: Application to NFC Payment. To appear in the proceedings of Financial Cryptography'15.

A Useful Bounds

We recall here some useful bound on the tail of the binomial distribution.

Lemma 18 (Chernoff-Hoeffding bound [14,24]). *For any* ε, n, τ, q *such that* $\frac{\tau}{n} < q - \varepsilon$*, we have* $\mathsf{Tail}(n,\tau,q) > 1 - e^{-2\varepsilon^2 n}$*. For* $\frac{\tau}{n} > q + \varepsilon$*, we have* $\mathsf{Tail}(n,\tau,q) < e^{-2\varepsilon^2 n}$*.*

B Definitions

Definition 19 (Bit commitment). *A bit commitment consists of a PPT algorithm* Com *taking as input* λ*, a bit* $b \in \mathbf{Z}_2$*, and some random* $\rho \in G$*. It computes* $\mathsf{Com}(b;\rho) \in G$*. We define the following properties:*

- *homomorphic: for all* $b, b' \in \mathbf{Z}_2$ *and* $\rho, \rho' \in G$*,* $\mathsf{Com}(b;\rho)\mathsf{Com}(b';\rho') = \mathsf{Com}(b+b';\rho\rho')$*;*
- *perfectly binding: for all* $b, b' \in \mathbf{Z}_2$ *and* $\rho, \rho' \in G$*,* $\mathsf{Com}(b;\rho) = \mathsf{Com}(b';\rho')$ *implies* $b = b'$*;*
- *computationally hiding: for* ρ *random, the distributions of* $\mathsf{Com}(0;\rho)$ *and* $\mathsf{Com}(1;\rho)$ *are computationally indistinguishable.*

Definition 20 (Sound proof of membership). *An interactive proof for a language* L *is a pair of protocols* $(P(\zeta), V(z))$ *of PPT algorithms such that*

- completeness*: for any* $z \in L$ *with witness* ζ*,* $\Pr[\mathsf{Out}_V = 1 : P(\zeta) \leftrightarrow V(z)] = 1$*;*
- κ-soundness*: for any* $z \notin L$ *and any algorithm* P^* *then* $\Pr[\mathsf{Out}_V = 1 : P^* \leftrightarrow V(z)] \leq \kappa$*.*

Definition 21 (Zero-knowledge protocol). *A protocol* $(P(\zeta), V(z))$ *for a language* L *is computationally zero-knowledge for* $P(\zeta)$ *if for any PPT interactive machine* $V^*(z, \mathsf{aux})$ *there exists a PPT algorithm* $S(z, \mathsf{aux})$ *and a negligible* ε *such that for any PPT distinguisher, any* $(z : \zeta) \in L$*, and any* aux*, the advantage for distinguishing the final view of* $V^*(z, \mathsf{aux})$ *in* $P(\zeta) \leftrightarrow V^*(z, \mathsf{aux})$ *and the output of* $S(z, \mathsf{aux})$ *is bounded by* ε*.*

Definition 22 (One-way function). *We consider a function* Com *taking as input* λ *and a message* $x \in \mathbf{Z}_2^s$ *which is computable in deterministic polynomial time. The function is one-way if for any algorithm receiving* $\mathsf{Com}(x)$*, for* $x \in \mathbf{Z}_2^s$ *random, the probability that it outputs* x *is negligible.*

C VSSDB

At BalkanCryptSec'14, Gambs *et al.* presented VSSDB, a new protocol based on the random oracle model.[7] The protocol also needs a PKI for the verifier, so that he can use a secret decryption key sk_v and publish a public encryption key pk_V. It also uses a homomorphic bit-commitment scheme. The presented protocol offers no tolerance to noise and is not sound, but it offers resistance to terrorist fraud as modeled by the GameTF notion [19] in the DFKO model.

[7] http://www.gstl.itu.edu.tr/BalkanCryptSec/presentations/Gambs.ppt

Verifier		**Prover**
public: z	$z = \zeta^2$	secret: ζ
	$\xleftarrow{\mathsf{pk}}$	generate sk, $\mathsf{pk} = \mathsf{Gen}(\mathsf{sk})$
pick $e \in \mathbf{Z}_2$, pick r	$\xrightarrow{\mathsf{Commit}_{\mathsf{pk}}(e;r)}$	
	$\xleftarrow{h}$	pick $g \in G$, $h = g^2$
	$\xrightarrow{e,r}$	open commitment
check $z^e = \ell^2 h^{-1}$, $\mathsf{pk} = \mathsf{Gen}(\mathsf{sk})$	$\xleftarrow{\ell,\mathsf{sk}}$	$\ell = g\zeta^e$
	$\xrightarrow{\mathsf{Out}_V}$	

Fig. 3. $\mathsf{ZKP}(z : \zeta)$: a Sound and Zero-Knowledge Proof for z Being a Square based on a trapdoor commitment Commit.

Verifier		**Prover**
public: y		secret: x
	initialization phase	
		pick $a \in \mathsf{GF}(q)^n$
	$\xleftarrow{A}$	$A = \mathsf{Commit}(a;\rho)$
	challenge phase	
	for $i = 1$ to n	
pick $c_i \in \mathsf{GF}(q)$		
start timer_i	$\xRightarrow{c_i}$	receive c'_i
receive r_i, stop timer_i	$\xLeftarrow{r'_i}$	$r'_i = a_i + c'_i$
	verification phase	
$\mathsf{Verify}_y(\sigma, \mathsf{transcript})$, verify $A = \mathsf{Commit}(a;\rho)$	$\xleftarrow{a,\rho,\sigma,c',r'}$	$\sigma = \mathsf{Sign}_x(\mathsf{transcript})$
check $\#\{i; \mathsf{timer}_i \le 2B, c_i = c'_i, r_i = r'_i = a_i + c_i\} \ge \tau$	$\xrightarrow{\mathsf{Out}_V}$	

Fig. 4. The Brands-Chaum Protocol [12].

Verifier		**Prover**
secret key: x_V		secret key: x_P
public key: $y_V = x_V G$		public key: $y_P = x_P G$
	initialization phase	
		pick $r_1, r_2 \in \mathbf{Z}_\ell^*$
pick $r_3, e \in \mathbf{Z}_\ell^*$	$\xleftarrow{R_1, R_2}$	$R_1 = r_1 G$, $R_2 = r_2 G$
$R_3 = r_3 G$	$\xrightarrow{R_3}$	
$a_0 \| a_1 = \mathsf{xcoord}_{2n}(r_3 R_1)$		$a'_0 \| a'_1 = \mathsf{xcoord}_{2n}(r_1 R_3)$
	challenge phase	
	for $i = 1$ to n	
$c_i = \mathsf{bit}_i(e)$		
start timer_i	$\xrightarrow{c_i}$	
stop timer_i	$\xleftarrow{f_i}$	$f_i = a'_{c_i,i}$
	verification phase	
check $\mathsf{timer}_i \leq 2B$	$\xrightarrow{e}$	check $\mathsf{bit}_i(e) = c_i$
check $f_i = a_{c_i,i}$	$\xleftarrow{s}$	$s = x_P + e r_1 + r_2 + \mathsf{xcoord}(r_2 y_V)$
validate $y_P = (s - \mathsf{xcoord}(x_V R_2))G - e R_1 - R_2$	$\xrightarrow{\mathsf{Out}_V}$	
private output: y_P		

Fig. 5. The Hermans-Peeters-Onete Protocol [23].

The security proof of VSSDB is not available at the time of writing. However, we can see that it is hard to create a NIZK proof for that c_P being well formed, since it depends on outputs from the random oracle H which must remain hidden. Namely, we cannot have any proof that $H(\nu_{i-1}) = \nu_i$ without disclosing ν_{i-1} and letting the verifier access H to retrieve ν_i.

Verifier		**Prover**
secret key: sk_V		secret key: x, sk_P
public key: pk_V		public key: $\mathsf{Com}, \mathsf{pk}_P$
		$\mathsf{Com}_i = \mathsf{Commit}(x_i; H^i(x))$, $i = 1, \ldots, m$

initialization phase

(in the next two messages, if a participant receives a first bit set to 1, the protocol escapes as in [19])

pick N_V	$\xrightarrow{0 \Vert N_V}$	for $i = 1, \ldots, m$:
		pick $k_i, \ell_i \in \{0,1\}$, u_i, v_i
		$e_i = k_i \oplus \ell_i \oplus x_i$
		$w_i = H^i(x) u_i^{-1} v_i^{-1}$
		$a_i = \mathsf{Commit}(k_i; u_i)$
		$b_i = \mathsf{Commit}(\ell_i; v_i)$
		$d_i = \mathsf{Commit}(e_i; w_i)$
		$m_P = (a_i, b_i, d_i)_{i=1,\ldots,m} \Vert N_V$
		$c_P = \mathsf{Enc}_{\mathsf{pk}_V}(m_P \Vert \mathsf{Sign}_{\mathsf{sk}_P}(m_P))$
$m' \Vert N_V^* \Vert \sigma^* = \mathsf{Dec}_{\mathsf{sk}_V}(c_P)$	$\xleftarrow{0 \Vert c_P, \pi}$	π: proof that c_P is well formed and consistent with Com
verify π, σ^*, $N_V = N_V^*$		
pick $M \in \{0,1\}^m$	$\xrightarrow{M}$	receive $\hat{M}$

challenge phase

for $i = 1$ to n

pick $c_i \in \{0,1\}$, start timer_i	$\xrightarrow{c_i}$	receive $\hat{r}_i$
		$f(0,0) = e_i$ $\quad \gamma(0,0) = w_i$
		$f(0,1) = k_i \oplus \ell_i$ $\quad \gamma(0,1) = (k_i, u_i, \ell_i, v_i)$
		$f(1,0) = k_i$ $\quad \gamma(1,0) = u_i$
		$f(1,1) = \overline{e_i \oplus \ell_i}$ $\quad \gamma(1,1) = (e_i, w_i, \ell_i, v_i)$
stop timer_i	$\xleftarrow{r_i}$	$r_i = f(\hat{M}_i, \hat{c}_i)$, $\gamma_i = \gamma(\hat{M}_i, \hat{c}_i)$

verification phase

verify σ	$\xrightarrow{\gamma, \sigma}$	$\sigma = \mathsf{Sign}_{\mathsf{sk}_P}(M \Vert c \Vert r \Vert N_V)$
for $i = 1, \ldots, m$:		
check $\mathsf{timer}_i \leq 2B$, $a_i, b_i, d_i, \mathsf{Com}_i$ consistent with r_i, γ_i	$\xrightarrow{\mathsf{Out}_V}$	

Fig. 6. VSSDB.