Sherlock Holmes Zero-Knowledge Protocols

George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
tgeorge@dcti.ro
[2] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania

Abstract. We present two simple zero knowledge interactive proofs that can be instantiated with many of the standard decisional or computational hardness assumptions. Compared with traditional zero knowledge proofs, in our protocols the verifiers starts first, by emitting a challenge, and then the prover answers the challenge.

1 Introduction

A standard interactive proof of knowledge involves a prover, usually called P or $Peggy$, and a verifier, usually called V or $Victor$. $Peggy$ is in possession of some secret k and by interacting with $Victor$ she wants to convince him that she indeed owns k. More formally, an interactive proof is a pair of programs that implement the protocol between $Peggy$ and $Victor$. To be useful, such a proof must be complete and sound. By complete we mean that an honest $Peggy$ succeeds in convincing an honest $Victor$ and by sound we mean that a dishonest prover does not succeed in convincing the verifier of a false statement. Moreover, if $Victor$ does not learn anything from the protocol's execution which he did not know before, we say that the protocol is zero knowledge.

In a classical zero knowledge protocol, $Peggy$ starts the protocol by sending a commitment to $Victor$, then $Victor$ sends a challenge to $Peggy$ and finally $Peggy$ sends her answer. The verifier will accept the proof if and only if $Peggy$'s answer coincides with the answer he expects. In contrast with these protocols, the authors of [10] introduce a new class of protocols in which $Victor$ starts the protocol. Once the verifier knows that $Peggy$ wants to start the protocol[3], he issues a challenge to which $Peggy$ answers. If the answer is correct, then the protocol ends successfully. Otherwise, it fails.

Although Grigoriev and Shpilrain's protocol is very interesting, the authors only claim that their protocol is zero knowledge without actually proving it. To fill this gap, we re-formalized and generalized Grigoriev and Shpilrain's protocol and then we proved its security. A downside of this formalization, is that

[3] *e.g.* $Peggy$ can send a "hello" type message or $Victor$ can be equipped with motion sensors and detect $Peggy$'s proximity

$Victor$ must iterate the protocol a number of times in order to fulfill the soundness property. By vectorizing the protocol we managed to reduce the number of iteration to one.

To further improve our protocol, we modified it by changing the underlying assumption from a decisional one to a computational one. This was necessary in order to reduce the bandwith requirements necessary for the decisional version. Note that if $Peggy$ and $Victor$ choose the right parameters the new protocol will provide the same security assurances.

Finally, we offer the reader several concrete realizations of our protocols and compare them with classical zero knowledge protocols such as Schnorr [17], Guillou-Quisquater [11] and Fiat-Shamir [7]. Note that one can devise new instantiations of our protocols.

Structure of the paper. We introduce notations and definitions used throughout the paper in Section 2. Inspired by Grigoriev and Shpilrain's protocol, in Section 3 we formalize and analyse the Multi-Decisional Sherlock Holmes (MDSH) protocol. A vectorized version of MDSH is presented in Section 4 and a computational version is tackled in Section 5. Section 6 contains a comparison with classical zero knowledge protocols. We conclude in Section 7.

2 Preliminaries

Notations. Throughout the paper, the notation $|S|$ denotes the cardinality of a set S. The action of selecting a random element x from a sample space X is denoted by $x \xleftarrow{\$} X$, while $x \leftarrow y$ represents the assignment of value y to variable x. The probability of the event E to happen is denoted by $Pr[E]$. The subset $\{0, \ldots, s-1\} \in \mathbb{N}$ is denoted by $[0, s]$. A vector v of length n is denoted either $v = (v_0, \ldots, v_{n-1})$ or $v = \{v_i\}_{i \in [0,n]}$ and $v_1 = v_2$ stands for element-wise equality between two vectors v_1 and v_2.

2.1 Hardness Assumptions

Inspired by the computational and decisional hardness assumptions described in [2] and the one way function definitions found in [1,16], we further provide the reader with the following two definitions. The first one captures the idea of a generic computational hardness assumption, while the second the decisional version. We do not claim to capture all the generic hardness assumptions, but for our purpose these definitions suffice. Note that when we define an advantage, we use ";" to denote the end of simple instructions or for loops and "," to denote the end of an instruction inside a for loop.

Definition 1 (Computational Hardness Assumption). *Let $K \subseteq \{0,1\}^*$ be a family of indices and for $k \in K$ let $D_k, R_k \subseteq \{0,1\}^*$. A computational hard function f is a parameterized family of functions $f_k : D_k \to R_k$ such that*

1. *for every $k \in K$ there exists a PPT algorithm that on input $x \in D_k$ outputs $f_k(x)$;*
2. *for every PPT algorithm A the advantage*

$$ADV_f^{CHA}(A) = Pr[f_k(z) = y \mid k \xleftarrow{\$} K; x \xleftarrow{\$} D_k; y \leftarrow f_k(x); z \leftarrow A(f_k, y)]$$

   is negligible;
3. *there exists a PPT algorithm B such that*

$$Pr[f_k(z) = y \mid k \xleftarrow{\$} K; x \xleftarrow{\$} D_k; y \leftarrow f_k(x); z \leftarrow B(k, y)] = 1.$$

Definition 2 (Decisional Hardness Assumption). *A function f is a decisional hard function if in Definition 1, Item 2 and 3 are changed to*

2. *for every PPT algorithm A the advantage*

$$ADV_f^{DHA}(A) = |2Pr[b = b' \mid k_0, k_1 \xleftarrow{\$} K; b \xleftarrow{\$} \{0,1\}; \\ x \xleftarrow{\$} D_{k_b}; y \leftarrow f_{k_b}(x); b' \leftarrow A(f_{k_0}, f_{k_1}, y)] - 1|$$

   is negligible;
3. *there exists a PPT algorithm B such that*

$$Pr[b = b' \mid k_0, k_1 \xleftarrow{\$} K; b \xleftarrow{\$} \{0,1\}; \\ x \xleftarrow{\$} D_{k_b}; y \leftarrow f_{k_b}(x); b' \leftarrow B(k_0, k_1, y)] = 1.$$

2.2 Zero-Knowledge Protocols

Let $Q : \{0,1\}^* \times \{0,1\}^* \to \{\texttt{true}, \texttt{false}\}$ be a predicate. Given a value z, Peggy will try to convince Victor that she knows a value x such that $Q(z, x) = \texttt{true}$.

We further base our reasoning on both a definition from [6,12] and a definition from [9,12] which we recall next.

Definition 3 (Proof of Knowledge Protocol). *An interactive protocol (P, V) is a proof of knowledge protocol for predicate Q if the following properties hold*

- **Completeness***: V accepts the proof when P has as input a value x with $Q(z, x) = \texttt{true}$;*
- **Soundness***: there exists an efficient program K (called knowledge extractor) such that for any $\bar{P}$ (possibly dishonest) with non-negligible probability of making V accept the proof, K can interact with $\bar{P}$ and output (with overwhelming probability) an x such that $Q(z, x) = \texttt{true}$.*

Definition 4 (Zero Knowledge Protocol). *A protocol (P, V) is zero-knowledge if for every efficient program $\bar{V}$ there exists an efficient program S, the simulator, such that the output of S is indistinguishable from a transcript of the protocol execution between P and $\bar{V}$.*

Remark 1. Note that we further work in the honest verifier scenario.

3 Multi-Decisional Protocol

3.1 Description

Based on a variation of decisional hard functions, we further describe a protocol (see Figure 1) that allows $Peggy$ to prove to $Victor$ that she is in possession of some secrets. When $Victor$ knows that $Peggy$ is ready to start the protocol, he sends her a challenge and $Peggy$ responds with her guess. If the guess is correct, then $Victor$ accepts the answer.

$Peggy$		$Victor$
Knows k_i, for $i \in [0, n]$		Knows f_{k_i}, for $i \in [0, n]$ Choose $i \xleftarrow{\$} [0, n]$ Choose $x \xleftarrow{\$} D_{k_i}$ Compute $y \leftarrow f_{k_i}(x)$
	$\xleftarrow{\quad y \quad}$	
$i' \leftarrow -1$ For $j \in [0, n]$ If $y \in R_{k_j}$ then $i' \leftarrow j$ If $i' = -1$ then abort		
	$\xrightarrow{\quad i' \quad}$	
		If $i' = i$ return `true` Else return `false`

Fig. 1. Multi-Decisional Sherlock Holmes (MDSH) Protocol.

Remark 2. The probability of an adversary guessing the correct index i is $1/n$. Thus, the protocol must be repeated sufficient number of times (*e.g.* m times) in order to prevent an attacker[4] to convince $Victor$ that he knows k_i, for $i \in [0, n]$.

3.2 Security Analysis

To ease understanding, we first introduce the notion of a multi-decisional hard function and then we prove the security of the MDSH protocol. At the end of this, subsection we show how to relate the security of a multi-decisional function to the security of a decisional function.

Definition 5 (Multi-Decisional Hardness Assumption). *Let $n \geq 2$ be an integer. A function f is a multi-decisional hard function if in Definition 2, Item 2 and 3 are changed to*

[4] In this case, the attacker's success probability is $1/n^m$.

2. *for every PPT algorithm A the advantage*

$$ADV_f^{\text{MDHA}}(A) = |2Pr[i = i' \mid \textit{for } i \in [0,n] : k_i \xleftarrow{\$} K; i \xleftarrow{\$} [0,n]; x \xleftarrow{\$} D_{k_i};$$
$$y \leftarrow f_{k_i}(x); i' \leftarrow A(f_k, y)] - 1|$$

is negligible, where $f_k = \{f_{k_i}\}_{i \in [0,n]}$*;*

3. *there exists a PPT algorithm B such that*

$$Pr[i = i' \mid \textit{for } i \in [0,n] : k_i \xleftarrow{\$} K; i \xleftarrow{\$} [0,n]; x \xleftarrow{\$} D_{k_i};$$
$$y \leftarrow f_{k_i}(x); i' \leftarrow B(k, y)] = 1,$$

where $k = \{k_i\}_{i \in [0,n]}$.

Remark 3. Please be advised that in the case of the multi-decisional hardness assumption we implicitly assume that all the keys are kept secret and none of them are leaked to an adversary (dishonest prover). If, for example, t out of n keys are leaked there is a simple strategy that makes the attacker win with probability $(t+1)/n$. More precisely, his strategy works as follows: The attacker, upon receipt of the verifier's challenge y, checks whether the message belongs to the set R_{k_i} for any of the t known secrets. If true (that happens with probability t/n), the attacker correctly answers the corresponding index of the matching secret. Otherwise, the attacker answers a random index chosen among the unknown secrets. In this last case, the success probability is $1/(n-t) \cdot (n-t)/n = 1/n$. Hence, the total success probability is $t/n + 1/n = (t+1)/n$.

Theorem 1. *The MDSH protocol is a proof of knowledge if and only if f is a multi-decisional hard function. Moreover, the protocol is zero knowledge.*

Proof. If f is a multi-decisional hard function, then according to Definition 5, Item 3, Peggy will compute with probability 1 the correct index. Thus, the completeness property is satisfied.

Let $\tilde{P}$ be a PPT algorithm that takes as input $f_{k_0}, \ldots, f_{k_{n-1}}$ and makes V accept the proof with non-negligible probability $Pr(\tilde{P})$. Then we are able to construct a PPT algorithm Q (described in Algorithm 1) that interacts with $\tilde{P}$ and that has a non-negligible advantage $ADV_f^{\text{MDHA}}(Q) = Pr(\tilde{P})$. Thus, the soundness property is satisfied.

Algorithm 1. Algorithm Q.

Input: An element $y \leftarrow f_{k_i}(x)$ and n functions f_{k_i}, where $i \in [0,n]$
1. Send y to $\tilde{P}$
2. Receive i' from $\tilde{P}$
3. **return** i'

The last part of our proof consists in constructing a simulator S such that its output is indistinguishable from a genuine transcript between *Peggy* and *Victor*. Such a simulator is described in Algorithm 2.

Algorithm 2. Simulator S.

Input: n functions f_{k_i}, where $i \in [0, n]$

Choose $i \xleftarrow{\$} [0, n]$

Choose $x \xleftarrow{\$} D_{k_i}$

Compute $y \leftarrow f_{k_i}(x)$

return (y, i)

$\square$

We further show that if ADV_f^{DHA} is negligible, then MDSH is secure. Thus, when instantiating MDSH it suffices to know that decisional functions are secure.

Theorem 2. *For any PPT algorithm A there exists a PPT algorithm B such that the following inequality holds*

$$ADV_f^{\mathit{MDHA}}(A) \leq ADV_f^{\mathit{DHA}}(B).$$

Proof. Let A have a non-negligible advantage $ADV_f^{\text{MDHA}}(A)$. We describe in Algorithm 3 how B can obtain a non-negligible advantage $ADV_f^{\text{DHA}}(B)$ by interacting with A. Note that we have to randomly shuffle the functions' positions, in order to ensure that the index is randomly chosen from $[0, n]$.

Algorithm 3. Algorithm B.

Input: An element $y \leftarrow f_{k_b}(x)$

, where $b \xleftarrow{\$} \{0, 1\}$ **for** $i \in [2, n]$ **do**

  Choose $k_i \xleftarrow{\$} K$

end

Randomly shuffle $f_{k_0}, \ldots, f_{k_{n-1}}$'s positions and denote the result by $f'_{k_0}, \ldots, f'_{k_{n-1}}$

Let $i' \leftarrow A(f'_{k_0}, \ldots, f'_{k_{n-1}}, y)$

if i' *is the position of* f_{k_0} **then**

  return 0

end

else if i' *is the position of* f_{k_1} **then**

  return 1

end

else

  return $\perp$

end

$\square$

3.3 Examples

Quadratic Residuosity Assumption. Let N be the product of two large primes p and q and let $J_N(x)$ denote the Jacobi symbol of x modulo N. We denote by $J_N = \{x \in \mathbb{Z}_N^* \mid J_N(x) = 1\}$ and $QR_N = \{x \in \mathbb{Z}_N^* \mid J_p(x) = 1 \text{ and } J_q(x) = 1\}$. Let u be an element such that his Jacobi symbol $J_N(u)$ is 1. The *quadratic residuosity assumptions* (denoted by QR) states that deciding if $u \in J_N \setminus QR_N$ or $u \in QR_N$ is intractable without knowing p or q (see [5]).

Since QRA partitions J_N in two sets, we must set $n = 2$ for MDSH. Let u be an element such that $J_p(u) = J_q(u) = -1$. Then the MDSH parameters are as follows

- the secret keys are $k_0 = k_1 = (p, q)$;
- the functions are defined as $f_{k_0}(x) = x^2 \bmod N$ and $f_{k_1}(x) = u \cdot x^2 \bmod N$, where u and N are public.

To decide if $y \in J_N \setminus QR_N$ or $y \in QR_N$, *Peggy* computes $J_p(y)$. Note that when $b = 0$ we have $J_p(y) = J_p(x^2) = 1$ and when $b = 1$ we have $J_p(y) = J_p(u)J_p(x^2) = -1$.

Remark 4. A similar assumption can be found in [3]. Let $\kappa > 1$ be an integer and let $p, q \equiv 1 \bmod 2^\kappa$. Then the *gap* 2^κ*-residuosity assumption* states that is hard to distinguish between an element from $J_N \setminus QR_N$ and element of the form $y^{2^\kappa} \bmod N$, where $y \in \mathbb{Z}_N^*$. In this case the functions become $f_{k_0}(x) = x^{2^\kappa} \bmod N$ and $f_{k_1}(x) = u \cdot x^{2^\kappa} \bmod N$

Least Significant Bit of the e-th Root Assumption. Let $N = pq$ be the product of two large primes. We denote by $\varphi(N)$ the Euler totient function. Let e be an integer such that $gcd(e, \varphi(N)) = 1$. The *least significant bit of the e-th root assumption* (denoted LSB-ER) states that given $y \equiv x^e \bmod N$ is hard to decide if the least-significant bit of x is 0 or 1 (see [15]).

As in the case of QR, we have $n = 2$. The protocol's parameters are

- the secret keys are $k_0 = k_1 = (p, q)$;
- the functions are defined as $f_{k_0}(x) = (2x)^e \bmod N$ and $f_{k_1}(x) = (2x+1)^e \cdot x^2 \bmod N$, where N and e are public.

To find the least significant bit lsb, *Peggy* computes a d such that $ed \equiv 1 \bmod \varphi(N)$ and an element $z \leftarrow y^d \bmod N$. Then $lsb \equiv z \bmod 2$.

Decisional Diffie-Hellman Assumption. Let $\mathbb{G}$ be a cyclic group of prime order q and g a generator of $\mathbb{G}$. Let $x_1, x_2, y \xleftarrow{\$} \mathbb{Z}_q^*$ and $b \xleftarrow{\$} \{0, 1\}$. The *decisional Diffie-Hellman assumption* (denoted by DDH) states that given $(g^{x_1}, g^{x_2}, g^y, (g^{x_b})^y)$ the probability for a PPT algorithm to compute the bit b is negligible (see [2]).

In this case $n \geq 2$ and the parameters are

- the secret keys are $k_i \xleftarrow{\$} \mathbb{Z}_q^*$, for $i \in [0, n]$;

– the public parameters are $r_i \leftarrow g^{k_i}$, for $i \in [0, n]$, the group $\mathbb{G}$ and the generator g;
– the functions are defined as $f_{k_i}(x) = (g^x, r_i^x)$, for $i \in [0, n]$.

To decide the correct index, $Peggy$ has to parse $y = (y_0, y_1)$ and to compute $\ell = y_0^{k_i}$ until $\ell = y_1$. Note that $y_0^{k_i} = r_i^x$.

Decisional Bilinear Diffie-Hellman Assumption. Let $\mathbb{G}$ be cyclic group of prime order q and let P be the corresponding generator. We denote by $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a cryptographic bilinear map, where G_T is a cyclic group of order q. We will use the convention of writing $\mathbb{G}$ additively and $\mathbb{G}_T$ multiplicatively.

Let $a_0, a_1, b_0, b_1, c \xleftarrow{\$} \mathbb{Z}_q^*$. The *decisional bilinear Diffie-Hellman assumption* (denoted DBDH) states that given $(a_0P, a_1P, b_0P, b_1P, cP, Z)$ the probability of deciding if $Z = e(P, P)^{a_0b_0c}$ or $Z = e(P, P)^{a_1b_1c}$ is negligible (see [4]).

As in the case of DDH, we have $n \geq 2$. The MDSH's parameters are

– the secret keys are $a_i, b_i \xleftarrow{\$} \mathbb{Z}_q^*$, for $i \in [0, n]$;
– the public parameters are $Q_i \leftarrow a_iP$ and $R_i \leftarrow b_iP$, for $i \in [0, n]$, the group $\mathbb{G}$, the generator P and the bilinear map e;
– the functions are defined as $f_{k_i}(x) = (xP, e(Q_i, R_i)^x)$, for $i \in [0, n]$.

To find the correct answer, $Peggy$ parses $y = (Y_0, Y_1)$ and computes $L = e(P, Y_0)^{a_ib_i}$ until $L = Y_1$. Note that $e(Q_i, R_i)^x = e(P, P)^{a_ib_ix} = e(P, xP)^{a_ib_i} = e(P, Y_0)^{a_ib_i}$.

4 Vectorized Multi-Decisional Protocol

4.1 Description

A downside to the MDSH protocol is that $Victor$ has to run the protocol a number of times before he can be sure that $Peggy$ knows $\{k_i\}_{i\in[0,n]}$. We further present a variation of MDSH (see Figure 2) that allows $Victor$ to run the protocol only once, if he chooses the right parameters. Let $t > 1$ be an integer.

Remark 5. The probability of an adversary guessing the correct index vector v is $1/n^t$. If n^t is sufficiently large, then a single execution of the protocol suffices. Otherwise, $Victor$ must rerun the protocol multiple times.

4.2 Security Analysis

As in Section 3.2, we first introduce the relevant hardness assumption, then we prove the security of the VDSH protocol and at the end we relate the new hardness assumption with the multi-dimensional hardness assumption.

Definition 6 (Vectorized Multi-Decisional Hardness Assumption). *Let $t > 1$ be an integer. A function f is a vectorized multi-decisional hard function if in Definition 5, Item 2 and 3 are changed to*

$Peggy$		$Victor$

Knows k_i, for $i \in [0, n]$		Knows f_{k_i}, for $i \in [0, n]$
		For $j \in [0, t]$
		  Choose $i_j \xleftarrow{\$} [0, n]$
		  Choose $x_j \xleftarrow{\$} D_{k_{i_j}}$
		  Compute $y_j \leftarrow f_{k_{i_j}}(x_j)$
		Let $y = (y_0, \ldots, y_{t-1})$

	$\xleftarrow{\quad y \quad}$	

For $s \in [0, t]$
  $i'_s \leftarrow -1$
  For $j \in [0, n]$
    If $y_s \in R_{k_j}$ then $i'_s \leftarrow j$
  If $i'_s = -1$ then abort
Let $v' = (i'_0, \ldots, i'_{t-1})$
	$\xrightarrow{\quad v' \quad}$	

		Let $v = (i_0, \ldots, i_{t-1})$ If $v' = v$ return `true` Else return `false`

Fig. 2. Vectorized Multi-Decisional Sherlock Holmes (VDSH) Protocol.

2. *for every PPT algorithm A the advantage*

$$ADV_f^{\text{VDHA}}(A) = |2Pr[v = v' \mid \textit{for } i \in [0, n] : k_i \xleftarrow{\$} K; \textit{for } j \in [0, t] : i_j \xleftarrow{\$} [0, n],$$
$$x_j \xleftarrow{\$} D_{k_{i_j}}, y_j \leftarrow f_{k_{i_j}}(x_j); v' \leftarrow A(f_k, y)] - 1|$$

is negligible, where $f_k = \{f_{k_i}\}_{i \in [0,n]}$, $v = \{i_j\}_{j \in [0,t]}$ and $y = \{y_j\}_{j \in [0,t]}$;

3. *there exists a PPT algorithm B such that*

$$Pr[v = v' \mid \textit{for } i \in [0, n] : k_i \xleftarrow{\$} K; \textit{for } j \in [0, t] : i_j \xleftarrow{\$} [0, n],$$
$$x_j \xleftarrow{\$} D_{k_{i_j}}, y_j \leftarrow f_{k_{i_j}}(x_j); v' \leftarrow B(k, y)] = 1,$$

where $k = \{k_i\}_{i \in [0,n]}$, $v = \{i_j\}_{j \in [0,t]}$ and $y = \{y_j\}_{j \in [0,t]}$.

Theorem 3. *The VDSH protocol is a proof of knowledge if and only if f is a vectorized multi-decisional hard function. Moreover, the protocol is zero knowledge.*

Proof. The proof is similar to Theorem 2 and thus we only provide a sketch. The completeness property is satisfied due to Definition 6, Item 3.

A PPT algorithm R is described in Algorithm 4 and R has a non-negligible advantage $ADV_f^{\text{VDHA}}(R) = Pr(\tilde{P})$.

Finally, the simulator T is described in Algorithm 9

□

Algorithm 4. Algorithm R.

Input: A vector $y \leftarrow (f_k(x_0), \dots, f_k(x_{t-1}))$

Send y to $\tilde{P}$

Receive v' from $\tilde{P}$

return v'

Algorithm 5. Simulator T.

Input: n functions f_{k_i}, where $i \in [0, n]$

for $j \in [0, t]$ **do**

  Choose $i_j \xleftarrow{\$} [0, n]$

  Choose $x_j \xleftarrow{\$} D_{k_{i_j}}$

  Compute $y_j \leftarrow f_{k_{i_j}}(x)$

end

Let $y = (y_0, \dots, y_{t-1})$ and $v = (i_0, \dots, i_{t-1})$

return (y, v)

The next theorem proves the equivalence between the security notion associated with multi-decisional functions and the vectorized version of it. Using Theorems 2 and 4, the security of VDSH reduces to making sure that the decisional security notion is intractable.

Theorem 4. *For any PPT algorithms A and C there exist PPT algorithms B and D such that the following inequalities hold*

$$ADV_f^{MDHA}(A) \leq ADV_f^{VDHA}(B)$$
$$ADV_f^{VDHA}(C) \leq ADV_f^{MDHA}(D).$$

Proof. Let A have a non-negligible advantage $ADV_f^{\mathrm{MDHA}}(A)$ and let $Pr(A) = (ADV_f^{\mathrm{MDHA}}(A) + 1)/2$. We describe in Algorithm 6 how B can obtain a non-negligible advantage $ADV_f^{\mathrm{VDHA}}(B) = |2Pr(A)^n - 1|$ by interacting with A.

Algorithm 6. Algorithm B.

Input: A vector of elements $y \leftarrow (y_0, \dots, y_{t-1})$

for $j \in [0, t]$ **do**

  Let $i'_j \leftarrow A(f_{k_0}, \dots, f_{k_{n-1}}, y_j)$

end

Let $v' = (i'_0, \dots, i'_{t-1})$

return v'

To prove the second inequality we assume that $ADV_f^{\mathrm{VDHA}}(C)$ is non-negligible. Using algorithm C, we construct algorithm D (see Algorithm 7) that has a non-negligible advantage $ADV_f^{\mathrm{MDHA}}(D)$.

Algorithm 7. Algorithm D.

Input: An element $y \leftarrow f_{k_i}(x)$, where $i \xleftarrow{\$} [0, n]$

for $j \in [1, t]$ **do**
- Choose $i_j \xleftarrow{\$} [0, n]$
- Choose $x_j \xleftarrow{\$} D_{k_{i_j}}$
- Compute $y_j \leftarrow f_{k_{i_j}}(x)$

end

Let $z = (y, y_1, \ldots, y_{t-1})$ and $f_k = (f_{k_0}, \ldots, f_{k_{n-1}})$

Let $v' \leftarrow C(f_k, z)$

Parse $v' = (v'_0, \ldots, v'_{t-1})$

return v'_0

□

5 Computational Protocol

5.1 Description

Using a different security notion, we describe in Figure 3 a protocol that consumes less bandwith that the VDSH protocol, while maintaining its security, if the parameters are selected correctly.

Peggy		*Victor*
Knows k		Knows f_k Choose $x \xleftarrow{\$} D_k$ Compute $y \leftarrow f_k(x)$
	$\xleftarrow{\quad y \quad}$	
Compute z such that $f_k(z) = y$		
	$\xrightarrow{\quad z \quad}$	
		If $f_k(z) = y$ return `true` Else return `false`

Fig. 3. Computational Sherlock Holmes (CSH) Protocol.

Remark 6. The probability of an adversary guessing the correct element x is $1/|D_k|$. If $|D_k|$ is sufficiently large, then a single execution of the protocol suffices. Otherwise, the protocol must be repeated several times.

Remark 7. A vectorized version of the CSH protocol can also be constructed, but as we will see in Section 5.3 it is not necessary. Note that the security analysis is similar to the one from Section 4.2.

5.2 Security Analysis

Theorem 5. *The CSH protocol is a proof of knowledge if and only if f is a computational hard function. Moreover, the protocol is zero knowledge.*

Proof. The proof is similar to Theorem 2 and thus we only provide a sketch. The completeness property is satisfied due to Definition 1, Item 3.

A PPT algorithm O is described in Algorithm 8 and O has a non-negligible advantage $ADV_f^{\text{CHA}}(O) = Pr(\tilde{P})$. Note that in this case $\tilde{P}$ only takes as input a function f_k.

Algorithm 8. Algorithm O.

Input: An element $y \leftarrow f_k(x)$
1. Send y to $\tilde{P}$
2. Receive z from $\tilde{P}$
3. **return** z

Finally, the simulator U is described in Algorithm 9

Algorithm 9. Simulator U.

Input: A function f_k
1. Choose $x \xleftarrow{\$} D_k$
2. Compute $y \leftarrow f_k(x)$
3. **return** (y, x)

□

5.3 Examples

e-th Root Assumption. Using the same parameters as in the case of LSB-ER, the *e-th root assumption* (denoted ER) states that given $y \equiv x^e \bmod N$ computing x is intractable (see [12]).

Using this assumption we can instantiate the CSH protocol with $k = (p, q)$ and $f_k(x) = x^e \bmod N$. To recover x, *Peggy* has to compute a d such that $ed \equiv 1 \bmod \varphi(N)$ and then $x \leftarrow y^d \bmod N$.

Remark 8. The problem can also be stated for $e = 2$, but to find a solution to $x^2 \bmod N$, *Peggy* has to use a different technique (*e.g* the Shanks-Tonelli algorithm [13]). Note that this assumption is equivalent with the intractability of factoring N (*i.e. factoring assumption*).

Gap 2^κ-residuosity Assumption Using the same parameters as in Section 3.3, we can define $f_k(x) = y^x z^{2^\kappa} \bmod N$, where $k = (p, q)$, $D_k = [0, 2^\kappa]$ and $z \xleftarrow{\$} \mathbb{Z}_N^*$. A method for recovering x if one knows p is described in [3].

Computational Diffie-Hellman. Let $\mathbb{G}$ be a cyclic group of order q and g a generator of $\mathbb{G}$. Let $x, y \xleftarrow{\$} \mathbb{Z}_q^*$. The *computational Diffie-Hellman assumption* (denoted by CDH) states that given (g^x, g^y) is intractable to compute g^{xy} without knowing x or y (see [2]). In this case a more efficient version of the CSH protocol is provided in Figure 4.

Peggy		*Victor*
Knows x		Knows $y = g^x$ Choose $k \xleftarrow{\$} \mathbb{Z}_q^*$ Compute $r \leftarrow g^k$
	$\xleftarrow{\quad r \quad}$	
Compute $z \leftarrow r^x$		
	$\xrightarrow{\quad z \quad}$	
		If $z = y^k$ return `true` Else return `false`

Fig. 4. Diffie-Hellman Version of the CSH (DHCSH) Protocol.

Remark 9. Note that the DHCSH protocol was used in [19] to develop a method that performs full network authentication for resource-constrained devices. In [18], the authors introduce a version of the DHCSH protocol in which instead of sending r the verifier sends $(r, h(y^k))$, where h is a hash function. Stinson and Wu [18] prove that their protocol is secure against active intruders and reset attack[5]. A more efficient version of the Stinson-Wu protocol was introduced in [20,21]. In this variant, *Victor* sends r, while *Peggy* sends $h(z)$ instead of z. The authors [20,21] show that the scheme achieves the same security as their previously proposed protocol.

Computational Bilinear Diffie-Hellman Assumption. We assume the same setup as in the case of DBDH. Let $a, b, c \xleftarrow{\$} \mathbb{Z}_q^*$. The *computational bilinear Diffie-Hellman assumption* (denoted CBDH) states that given (aP, bP, cP) a PPT algorithm will compute $e(P, P)^{abc}$ with negligible probability (see [4]).

As in the case of CDH, this assumption allows us to have a more efficient version of the protocol. We will use Figure 4 as a reference. Thus, *Peggy* and *Victor* know $x = (a, b)$ and, respectively, $y = (aP, bP)$. The protocol's first step consists of *Victor* computing $r \leftarrow kP$. After that *Peggy* computes $z \leftarrow e(P, r)^{ab}$. Finally, the protocol's output is `true` if and only if $z = e(aP, bP)^k$.

[5] We refer the reader to [8] for a detailed description of these types of attacks.

6 Performance of the Sherlock Holmes Protocols

In this section we compare the Sherlock Holmes protocols to some classical zero knowledge protocols such as Schnorr [17], Guillou-Quisquater [11] and Fiat-Shamir [7].

We further assume the same setup as in the case of CDH. From Figure 5 we can see that the bandwidth requirement for Schnorr's protocol is $\log_2(|\mathbb{G}| + 2q)$ bits. Similarly, for the Diffie-Hellman version of the CSH protocol we obtain a requirement of $\log_2(2|\mathbb{G}|)$ bits. In practice, $\mathbb{G}$ is either $\mathbb{Z}_p^*$, where $p = (q-1)/2$ is a prime or an elliptic curve $E(\mathbb{Z}_p)$ such that $|E(\mathbb{Z}_p)| = hq$, where $h \leq 4$. Thus, in the modulo p case we obtain $(5q-1)/2$ versus $q-1$ and in the elliptic curve case $(h+2)q$ vs $2hq$. Thus, in most cases, our protocol's requirements are slightly lower. From a computational point of view, it is easy to see that both protocols have their complexity dominated by three exponentiations.

Peggy		*Victor*
Knows x		Knows $y = g^x$
Choose $k \xleftarrow{\$} \mathbb{Z}_q^*$		
Compute $r \leftarrow g^k$		
	$\xrightarrow{r}$	
		Choose $c \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xleftarrow{c}$	
Compute $s \leftarrow k + cx$		
	$\xrightarrow{s}$	
		If $g^s = ry^c$ return `true`
		Else return `false`

Fig. 5. Schnorr's Protocol.

Remark 10. Okamoto's protocol [14] can be seen as a vectorized version of Schnorr's protocol with $n = 2$. Thus, we can conclude that a vectorized version of DHCSH has slightly lower requirements as Okamoto's protocol.

Using Figure 5 as a reference, we further describe the Guillou-Quisquater (GQ) protocol. Assuming the setup from ER we set $y \equiv x^e \bmod N$. In the first phase, *Peggy* chooses $k \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $r \equiv k^e \bmod N$. Then *Victor* randomly selects $c \xleftarrow{\$} [0, e-1]$. The third step consists of *Peggy* computing $s \equiv kx^c \bmod N$. Then *Victor* accepts the proof if an only if $s^e \equiv ry^c \bmod N$.

The bandwidth requirement for the GQ protocol is $\log_2(2N+e)$, while for the e-th root instantiation of CSH is $loq_2(2N)$. Hence, the requirements are similar only if e is small. From a computational point of view, CSH's time is dominated by two exponentiations, while GQ's time by four. So, our protocol is twice as fast. Also, note that the probability of impersonating *Peggy* is $1/e$ for GQ, while for our protocol is in the worse case $e^2/\varphi(N)$[6].

The Fiat-Shamir protocol [7] considers $e = 2$. Let $n = 2$. If we consider $MDSH$ instantiated with DDH, we obtain a bandwith requirement of $\log_2(|\mathbb{G}|)$, a complexity dominated by three exponentiations and a probability of impersonating *Peggy* of $1/2$. Let $\mathbb{G} = \mathbb{Z}_{p'}^*$, when p' is prime[7]. Using the reasoning from the GQ protocol, we obtain that the $MDSH$ protocol has a better performance that the Fiat-Shamir, while having the same security.

7 Conclusions

Our two main zero knowledge protocols, decisional and computational Sherlock Holmes protocols, represent two new large classes of protocols. The presented list of examples is by no means exhaustive. Our next challenge is to see how we can adapt these protocols in order to obtain new cryptographic primitives (*e.g.* non-interactive zero knowledge proofs or digital signatures). Another interesting research direction is to investigate whether these protocols can be secured against active intruders and reset attack

References

1. Bellare, M., Goldwasser, S.: Lecture Notes on Cryptography. https://cseweb.ucsd.edu/~mihir/papers/gb.pdf (2008)
2. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography. https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf (2005)
3. Benhamouda, F., Herranz, J., Joye, M., Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. Journal of Cryptology **30**(2), 519–549 (2017)
4. Chatterjee, S., Sarkar, P.: Practical Hybrid (Hierarchical) Identity-Based Encryption Schemes Based on the Decisional Bilinear Diffie-Hellman Assumption. IJACT **3**(1), 47–83 (2013)
5. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: IMACC 2001. Lecture Notes in Computer Science, vol. 2260, pp. 360–363. Springer (2001)
6. Feige, U., Fiat, A., Shamir, A.: Zero-Knowledge Proofs of Identity. Journal of Cryptology **1**(2), 77–94 (1988)
7. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO 1986. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986)
8. Goldreich, O.: Zero-Knowledge Twenty Years After Its Invention. IACR Cryptology ePrint Archive **2002/186** (2002)
9. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. **18**(1), 186–208 (1989)

[6] According to Lagrange's theorem the polynomial x^e has at most e solution modulo p.

[7] In practice, for security reasons, n and p' have similar lengths.

10. Grigoriev, D., Shpilrain, V.: No-Leak Authentication by the Sherlock Holmes Method. Groups Complexity Cryptology **4**(1), 177–189 (2012)
11. Guillou, L.C., Quisquater, J.J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: EUROCRYPT 1988. Lecture Notes in Computer Science, vol. 330, pp. 123–128. Springer (1988)
12. Maurer, U.: Unifying Zero-Knowledge Proofs of Knowledge. In: AFRICACRYPT 2009. Lecture Notes in Computer Science, vol. 5580, pp. 272–286. Springer (2009)
13. Niven, I., Zuckerman, H.S., Montgomery, H.L.: An Introduction to the Theory of Numbers. John Wiley & Sons (1991)
14. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: CRYPTO 1992. Lecture Notes in Computer Science, vol. 740, pp. 31–53. Springer (1992)
15. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: PKC 2001. Lecture Notes in Computer Science, vol. 1992, pp. 104–118. Springer (2001)
16. Ostrovsky, R.: Foundations of Cryptography. `http://web.cs.ucla.edu/~rafail/PUBLIC/OstrovskyDraftLecNotes2010.pdf` (2010)
17. Schnorr, C.P.: Efficient Identification and Signatures For Smart Cards. In: CRYPTO 1989. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989)
18. Stinson, D.R., Wu, J.: An Efficient and Secure Two-Flow Zero-Knowledge Identification Protocol. Journal of Mathematical Cryptology **1**(3), 201–220 (2007)
19. Teşeleanu, G.: Lightweight Swarm Authentication. In: SECITC 2021. Lecture Notes in Computer Science, Springer (2021)
20. Wu, J., Stinson, D.R.: An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption. IACR Cryptology ePrint Archive **2007/479** (2007)
21. Wu, J., Stinson, D.R.: An Efficient Identification Protocol Secure Against Concurrent-Reset Attacks. Journal of Mathematical Cryptology **3**(4), 339–352 (2009)