# Efficient Authentication and Pseudorandomness from Weaker (Ring-)LPN Assumptions

Ivan Damgård[1], Sunoo Park[2], and Sarah Zakarias[1]

[1]Aarhus University
[2]MIT

**Abstract**

We propose a two new approaches to authentication based on the (ring-)LPN problem. In contrast to all known approaches, we can use a noise rate for the LPN problem that is arbitrarily close to 1/2, without this affecting the communication complexity of the protocol, and while doing only (poly-)logarithmic depth computation. At the cost of having the prover keep a small amount of state, our approach allows us to "upgrade" the HB protocol from passive to the man-in-the-middle security (the strongest notion) while maintaining its simple structure.

A technical contribution of independent interest is a construction of a poly-logarithmic depth PRF from LPN that is secure if at most a predetermined number $\ell$ of queries are asked; if more queries are asked, the same PRF is still secure, but now under a stronger assumption closely related to LPN. The basic idea of the construction also applies to other problems with a similar structure, such as subset-sum.

**Keywords.** LPN, ring-LPN, authentication, identification, pseudorandom functions.

## 1 Introduction

The well-known Learning Parity with Noise (LPN) problem involves a binary secret vector $\mathbf{s}$ of length $n$, and an adversary who is given a number of pieces of partial information about $\mathbf{s}$, which are called LPN samples. Each sample has form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a}$ is a uniformly random vector, $e$ is a bit that is 1 with probability $\tau$, and $\mathbf{s}$ is a random vector that is fixed across all samples. The adversary's goal is to compute $\mathbf{s}$ or (in the decision version of the problem) distinguish the samples from completely random.

Because of its simplicity and because it seems to be hard for constant $\tau < \frac{1}{2}$ and relatively small values of $n$, LPN has been intensively studied in order to build, for instance, efficient secret-key authentication protocols. That is, we have a prover and a verifier who have the same secret key $\mathbf{s}$, and we seek a protocol in which the verifier can make sure he is talking to the one prover who also knows the key. While the first suggestion (called HB [HB01]) only satisfied the most basic passive security requirement, later proposals have shown that we can get the strongest meaningful security for authentication protocols based on LPN [Kil+11], namely man-in-the-middle security. Later, the Ring-LPN problem was proposed as a way to further reduce the complexity of the protocols [Hey+12]. Here, samples are of form $a \cdot s + e$, where all three elements are in some suitable polynomial ring, and where each entry (i.e. coefficient) in $e$ is 1 with probability $\tau$.

From a theoretical point of view, one could also get secure authentication from LPN in a different way: there is a straightforward construction of a pseudorandom generator (PRG) from the LPN problem, and the result of [GGM86] shows how to build a pseudorandom function (PRF) from any PRG. Given a PRF $f_K$ that depends on a key $K$ shared by prover and verifier, consider a simple authentication protocol as follows: the verifier generates a random challenge $x$ and the

prover responds with $f_K(x)$. However this construction is rather inefficient: in particular, the PRF requires polynomial depth computation. In contrast, the other proposals for LPN-based authentication avoid this problem by using the special properties of LPN to do authentication without going via a PRF, and this allows them to make do with logarithmic depth computation.

Therefore, it may seem that the problem of doing authentication based on LPN is completely solved. However, all previous protocols share a somewhat unfortunate property, coming from the fact that the completeness of the protocols is not perfect. More precisely, we have to set the parameters in such a way that the honest verifier will reject the honest prover with only negligible probability. However at the same time, we also want to use parameters that make the underlying LPN problem instance as hard as possible, which clearly means choosing $\tau$ close to $\frac{1}{2}$. It turns out that if we want $\tau = \frac{1}{2} - \epsilon$ and error probability $2^{-w}$, then the communication in the protocol needs to be of size $\Omega(w/\epsilon^2)$. So as $\epsilon$ gets small, communication (and hence computation) grows quite dramatically. Using the PRF based construction via [GGM86] could potentially be used to avoid the growth in communication, but only at the cost of large depth computation. On the other hand, if we use a small value of $\tau$, the LPN problem gets easier, and we need to increase $n$ to maintain the security level, which again implies larger communication.

### 1.1 Our contribution

In this paper, we show we can allow an arbitrary value for $\tau$ (in particular, arbitrarily close to $\frac{1}{2}$) without having to pay for this with larger communication nor with large-depth computation. The price we pay is instead that the prover (but not the verifier) has to keep a small amount of state (basically a counter).

We show two approaches to the problem. The first one is a simple construction derived from the HB protocol. We show that if we replace the noise vectors generated by the prover by pseudorandom noise that the honest verifier can reconstruct, the protocol achieves perfect completeness and man-in-the-middle security, whereas the original HB protocol was only passively secure. The computation needed for the protocol is proportional to $\log(t)$ where $t$ is the number of times the protocol is used. Our protocol has the interesting property that its communication complexity depends only on the desired security level and not on $n$ or $\tau$. More precisely, if we want $k$-bit security, we need of course to choose parameters $(n, \tau)$ such that we can hope that the underlying LPN problem will take $2^k$ time to solve. But in our construction, this only affects the local computation of prover and verifier and not the communication, which always consists of 2 $k$-bit messages. This is in contrast to the one known protocol that is man-in-the-middle secure [Kil+11], where both computation and communication is affected by the choice of $n$ and $\tau$.

Our construction can also be seen as a general template for doing secure authentication based on a secure PRG. This is of particular interest for LPN because we know extremely efficient PRGs based on LPN, as explained later in the paper.

The second protocol is based on a PRF that we construct which can be computed in poly-logarithmic depth. When keys for the PRF are set up we choose a number $\ell$, and the PRF will then be secure if LPN is hard, provided the adversary asks at most $\ell$ queries ($\ell$ can be any polynomial in the security parameter). If more than $\ell$ queries are asked, this same PRF is still secure, but now under a stronger assumption that we describe in more detail below. The second protocol is secure if LPN is hard and its run time grows more slowly with $t$, namely it is proportional to $\log(\lceil t/\ell \rceil)$.

We believe that asking the prover to keep state is reasonable, and in particular, we avoid the synchrony problems that might arise if also the verifier had to keep state. Nevertheless, it is natural to ask what happens if the adversary is able to reset the prover. Fortunately, not all is lost: while the first protocol is insecure under such an attack, the second is secure, but now under the stronger assumption mentioned above.

It should be noted that although our construction pays a smaller price for driving $\tau$ towards $\frac{1}{2}$, we do have to pay in terms of computation (although the depth is small): the construction makes heavy use of the PRG that follows naturally from LPN, and this is less efficient if $\tau$ is close

to $\frac{1}{2}$. We believe that this is inherent, however, in the following sense: in all known protocols, security is based on arguing that the prover's responses are pseudorandom if LPN is hard (even for a malicious verifier). Now, if we consider a verifier that chooses his messages deterministically and interacts many times with the prover, then we see that the protocol is effectively a procedure that expands the prover's key into a long sequence of random looking messages (from the prover), that is, it acts as a PRG. This strongly suggests that we must build a PRG from LPN in order to get authentication, and hence suffer a loss in computational efficiency (if not in depth) as $\tau$ approaches $\frac{1}{2}$.

On the other hand, we could also choose to go for efficiency by choosing a relative small $\tau$ as this makes the stretch of the PRG larger. This requires us to increase $n$ to to maintain the concrete security level, and where existing protocols would suffer an increase in communication because of this, our communication complexity will not be affected.

It is a long standing open problem to construct a logarithmic depth PRF from LPN. Our work shows a first step towards a solution, that is already useful for the natural problem of authentication. As an observation of independent interest, we show that other problems that share some structure with LPN can be used as basis for our PRF construction: concretely, we show that the subset sum problem can be used in this way. Note that also for subset sum, it is not known how to construct a logarithmic depth PRF.

## 2 (Ring) Learning Parity with Noise

We begin by establishing some notation and formally defining the LPN [BFKL94] and ring-LPN [Hey+12] problems.

**Notation.** We denote vectors by bold lower-case letters, and matrices by bold capital letters. Where not stated otherwise, vectors are column vectors.

Let $\mathsf{Ber}_\tau$ denote the Bernoulli distribution with parameter $\tau$, and let $\mathsf{Ber}_\tau^n$ denote the distribution of vectors in $\mathbb{Z}_2^n$ where each bit is independently distributed according to $\mathsf{Ber}_\tau$. In the context of LPN, all arithmetic operations are modulo 2. For a polynomial ring $R \in \mathbb{F}_2[X]/(g)$, the distribution $\mathsf{Ber}_\tau^R$ denotes the distribution over $R$, where each of the coefficients of the polynomial is drawn independently from $\mathsf{Ber}_\tau$. For a polynomial $r \in R$, let $|r|$ denote the weight of $r$, i.e. the number of nonzero coefficients that $r$ has. Let $r[i] \in \mathbb{Z}_2$ denote the coefficient of $x^i$ in $r$. PPT stands for probabilistic polynomial time, $\mathsf{negl}(n)$ denotes a negligible function in $n$, and $\mathsf{poly}(n)$ denotes a function polynomial in $n$.

For a finite set $B$, we will use $U(B)$ to denote the uniform distribution over $B$. The relation $\overset{s}{\approx}$ between distributions denotes statistical indistinguishability, and $\overset{c}{\approx}$ denotes computational indistinguishability. $H$ is the binary entropy function.

**Definition 2.1** (Decisional LPN problem)**.** *Take parameters $n \in \mathbb{N}$ and $\tau \in \mathbb{R}$ with $0 < \tau \leq \frac{1}{2}$ (the noise rate). A distinguisher $\mathsf{D}$ is said to $(q, t, \varepsilon)$-solve the decisional $\mathsf{LPN}_{n,\tau}$ problem if*

$$\left| \Pr_{\mathbf{s},\mathbf{A},\mathbf{e}}[\mathsf{D}(\mathbf{A}, \mathbf{As} + \mathbf{e}) = 1] - \Pr_{\mathbf{r},\mathbf{A}}[\mathsf{D}(\mathbf{A}, \mathbf{r}) = 1] \right| \geq \varepsilon$$

*where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$, $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{q \times n}$, and $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_2^q$ are uniformly random and $\mathbf{e} \leftarrow \mathsf{Ber}_\tau^q$, and the distinguisher runs in time at most $t$.*

The decisional and search versions of the LPN problem are polynomially equivalent [KSS10].

The ring-LPN problem is a natural variant of the LPN problem with some additional structure allowing for fast multiplication and compact representations of samples, in which the vectors of the LPN problem are interpreted as polynomials in a ring.

**Definition 2.2** (Decisional ring-LPN problem)**.** *Take parameters $R = \mathbb{F}_2[X]/(g)$ with $g \in \mathbb{F}_2[X]$ irreducible and of degree $n-1$, and $\tau \in \mathbb{R}$ with $0 < \tau \leq \frac{1}{2}$. Let $\mathcal{U}^R$ denote the uniform distribution over $R \times R$. For any polynomial $s \in R$, let $\Lambda_\tau^{R,s}$ be the distribution over $R \times R$ whose samples are obtained by choosing a polynomial $r \xleftarrow{\$} R$ and another polynomial $e \to \mathsf{Ber}_\tau^R$ and outputting $(r, rs + e)$. A distinguisher $\mathsf{D}$ is said to $(q, t, \varepsilon)$-solve the decisional $\mathsf{RingLPN}_{n,\tau}^R$ problem if*

$$\left|\Pr[\mathsf{D}^{\Lambda_\tau^{R,s}} = 1] - \Pr[\mathsf{D}^{\mathcal{U}^R} = 1]\right| \geq \varepsilon$$

*and the distinguisher runs in time at most $t$ and makes at most $q$ queries.*

**Remark.** It is not strictly necessary (for correctness or security of the constructions) that $g$ be irreducible. In fact, it could increase efficiency of implementation if $g$ were not irreducible. However, there are various pitfalls to avoid when choosing a reducible $g$ (for example, it must not have factors of very low degree), in order to maintain security (see [Hey+12] for a more detailed discussion). In this work we assume that $g$ is irreducible.

Note that for both LPN and ring-LPN, when $\tau = \frac{1}{2}$, the samples are information-theoretically indistinguishable from uniformly random.

# 3 Overview of LPN-based authentication

A *authentication protocol* is an interactive two-party protocol $(\mathcal{P}, \mathcal{V})$ between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$: these may be respectively thought of as a (lightweight) *tag*, and a *reader* to which the tag is identifying itself. Both parties are PPT, and hold a shared secret $s$ generated according to some generation algorithm $\mathsf{Gen}(1^\kappa)$ (where $\kappa$ denotes the security parameter) in an initial phase. After an execution of the protocol, the verifier $\mathcal{V}$ outputs either accept or reject – this is also called the *output* of the protocol execution.

In this work we consider *prover-stateful protocols* where the prover also maintains some (small amount of) state between protocol executions, and *stateless* protocols where neither party needs to maintain state.

**Definition 3.1** (Completeness)**.** *The* completeness error *of a protocol is defined to be*

$$\Pr_{s \leftarrow \mathsf{Gen}(1^\kappa)}[(\mathcal{P}, \mathcal{V})(s) = \mathit{accept}].$$

*A protocol is* complete *if its completeness error is negligible in a security parameter. It is* perfectly complete *if its completeness error is zero.*

Common definitions of security for authentication protocols are given below. Note that the security definitions are presented in order of increasing strength, and the stronger security notions subsume the weaker ones.

**Definition 3.2** (Passive security)**.** *An authentication protocol $(\mathcal{P}, \mathcal{V})$ is* secure against passive attacks *if for any secret $s \leftarrow \mathsf{Gen}(1^\kappa)$, for any PPT adversary $\mathcal{A}$ which has access to arbitrarily polynomially many transcripts of honest protocol executions (for secret $s$), it holds that*

$$\Pr[(\mathcal{A}, \mathcal{V})(s) = \text{accept}] \leq \mathsf{negl}(\kappa).$$

**Definition 3.3** (Active security)**.** *An authentication protocol $(\mathcal{P}, \mathcal{V})$ is* secure against active attacks *if for any secret $s \leftarrow \mathsf{Gen}(1^\kappa)$, for any PPT adversary $\mathcal{A}$ which first can interact arbitrarily polynomially many times with an honest prover $\mathcal{P}$ (including concurrent executions, but not allowing resetting of the prover's state), and then afterward (now, without access to $\mathcal{P}$) interacts once with an honest verifier $\mathcal{V}$, it holds that*

$$\Pr[(\mathcal{A}, \mathcal{V})(s) = \text{accept}] \leq \mathsf{negl}(\kappa).$$

**Definition 3.4** (Man-in-the-middle (MIM) security)**.** *An authentication protocol* $(\mathcal{P}, \mathcal{V})$ *is* secure against man-in-the-middle (MIM) attacks *if for any* PPT *adversary* $\mathcal{A}$ *which first can interact arbitrarily polynomially many times with an honest prover* $\mathcal{P}$ *and/or an honest verifier* $\mathcal{V}$ *(including concurrent executions, but not allowing resetting of the prover's state), and then afterward (now, without access to* $\mathcal{P}$*) interacts once with an honest verifier* $\mathcal{V}'$*, it holds that*

$$\Pr[(\mathcal{A}, \mathcal{V}')(s) = \text{accept}] \leq \mathsf{negl}(\kappa).$$

*Note that in this setting,* $\mathcal{A}$ *learns the* accept/reject *decisions made by verifier* $\mathcal{V}$*.*

## 3.1 Overview of existing protocols

The first and simplest authentication scheme based on LPN was the HB scheme [HB01], illustrated in Protocol 1. HB is provably secure against passive attacks (but easily breakable by active attacks). Subsequently, [JW05] gave a variant protocol called HB$^+$ with an additional round (which requires the prover to keep state between rounds), and their protocol achieves active security as shown by [KSS10]. Then, [Kil+11] proposed the first two-round actively secure protocol, whose security is based on the Subspace LPN problem (this is a variant of the LPN problem that has been shown to be almost as secure as LPN itself, under certain conditions [PP03]).

**Public parameters.** Security parameter $\kappa \in \mathbb{Z}$, $n \in \mathbb{Z}$ with $n = \mathsf{poly}(\kappa)$, noise rate $\tau \in (0, \frac{1}{2})$, threshold $\tau' \in (\tau, \frac{1}{2})$.

**Key generation.** $\mathsf{Gen}(1^\kappa)$ samples secret key $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$.

| $\mathcal{P}(\mathbf{s})$ | | $\mathcal{V}(\mathbf{s})$ |
|---|---|---|
| | $\xleftarrow{\mathbf{A}}$ | $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{n \times m}$ |
| $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n$<br>$\mathbf{z} := \mathbf{As} + \mathbf{e}$ | $\xrightarrow{\mathbf{z}}$ | accept iff $\|\|\mathbf{z} + \mathbf{As}\|\|_1 < \tau' \cdot n$ |

Protocol 1: The HB authentication protocol

The preceding protocols are all vulnerable to man-in-the-middle attacks. [Kil+11] shows how to generate a MAC based on their aforementioned actively secure authentication protocol, and this yields the first efficient MIM secure protocol based on LPN (using two rounds). More recently, [DKPW12] gave a more efficient MIM secure variant protocol using pairwise independent hashing.

The notable authentication protocol in the literature which is presented in terms of ring-LPN rather than LPN is the Lapin protocol [Hey+12] (Protocol 2), which has provable active security. However, note that the preceding LPN-based protocols can also be straightforwardly adapted to the ring-LPN setting (the adaptation preserves security properties, but with respect to the ring-LPN assumption rather than the LPN assumption).

## 3.2 Increasing the noise rate

In existing schemes, though any constant noise rate $0 < \tau < \frac{1}{2}$ is supported, the completeness guarantees are asymptotic: in fact, as $\tau \to \frac{1}{2}$, the tradeoff between efficiency and completeness is rather poor. Concretely, for $\tau = \frac{1}{2} - \varepsilon$, if we want a completeness error of at most $e^{-w}$ for some fixed $w$, then by a Chernoff bound, $n = O(1/\varepsilon^2)$. Hence, to get arbitrarily low error probability we must have extremely large $n$ (and thus, very slow algorithms and large memory requirements). This is undesirable since we want $\tau$ close to $\frac{1}{2}$ in order to use the weakest assumption possible.

**Public parameters.** Security parameter $\kappa \in \mathbb{Z}$, $n \in \mathbb{Z}$ with $n = \mathsf{poly}(\kappa)$, noise rate $\tau \in (0, \frac{1}{2})$, threshold $\tau' \in (\tau, \frac{1}{2})$, polynomial ring $R = \mathbb{F}_2[X]/(g)$ with irreducible $g$ of degree $n$, and $\pi : \{0,1\}^\kappa \to R$ a mapping.

**Key generation.** $\mathsf{Gen}(1^\kappa)$ samples secret key $s \xleftarrow{\$} R$.

| $\mathcal{P}(\mathbf{s})$ | | $\mathcal{V}(\mathbf{s})$ |
|---|---|---|
| | $\xleftarrow{c}$ | $c \xleftarrow{\$} \mathbb{Z}_2^\kappa$ |
| $r \xleftarrow{\$} R, e \leftarrow \mathsf{Ber}_\tau^R$<br>$z := r \cdot (s \cdot \pi(c) + s') + e$ | $\xrightarrow{r,z}$ | $e' := z - r \cdot (s \cdot \pi(c) + s')$<br>accept iff $\|\|e'\|\|_1 < \tau' \cdot n$ |

Protocol 2: The Lapin authentication protocol

In contrast, by making use of (ring-)LPN-based pseudorandomness as in our protocols, we can greatly improve the tradeoff between efficiency and error rate.

# 4 Authentication using pseudorandom generation

It is known how to construct a simple and efficient pseudorandom generator based on LPN [BFKL94]. In this section we show how to improve authentication protocols using such a PRG.

**Definition 4.1** (Pseudorandom generator)**.** *Let $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a deterministic polynomial-time algorithm. $G$ is said to be a* pseudorandom generator *(PRG) if $m(n) > n$ and for any* PPT *algorithm $\mathcal{A}$ that outputs a single bit, it holds that $|\Pr[\mathcal{A}(r) = 1] - \Pr[\mathcal{A}(G(s)) = 1]| \leq \mathsf{negl}(n)$, where $r \leftarrow \{0,1\}^{m(n)}, s \leftarrow \{0,1\}^n$ are chosen uniformly at random, and the probabilities are taken over $r$, $s$, and the random coins of $\mathcal{A}$.*

It is well known that any pseudorandom generator implies pseudorandom generation with any polynomial expansion factor $m(n)$, by applying the PRG to its own output repeatedly.

## 4.1 (Ring-)LPN-based pseudorandom generation

The original pseudorandom generator of [BFKL94] takes a (uniformly random) input $\mathbf{r}$ and uses it to sample $\mathbf{A}$, $\mathbf{s}$ uniformly, and $\mathbf{e}$ with each bit distributed as $\mathsf{Ber}_\tau$, and then outputs $(\mathbf{A}, \mathbf{As} + \mathbf{e})$. Building upon this, [ACPS09] constructed an efficient (quasi-linear time) linear-stretch PRG based on the LPN problem. In particular, their construction makes use of the fact that $\mathbf{s} \in \mathbb{Z}_2^l$ and $\mathbf{e} \in \mathbb{Z}_2^m$ together have $l + m \cdot H(\tau)$ bits of entropy, and can be very efficiently sampled using roughly that many random bits by a method of [AIK08]. ($\mathbf{A}$ can be fixed as a public parameter and thus need not be sampled afresh each time.) However, the [ACPS09] construction is defined only for $\tau = 2^{-t}$ for $t \in \mathbb{Z}_+$. In Section 4.2, below, we generalize their construction to any real $\tau \in (0, \frac{1}{2}]$.

## 4.2 Efficient Bernoulli sampling, and PRGs for any noise rate

The PRF of [ACPS09] relies on the Bernoulli sampling technique of [AIK08], which is defined only for $\tau = 2^{-t}$ for $t \in \mathbb{Z}_+$.

**Lemma 4.2** ([AIK08])**.** *There exist positive integers $k > 1$ and $c > 2k$ and a sampling algorithm* $\mathsf{Samp}$ *that uses $(k + k/c)n$ random bits and outputs a pair $(\mathbf{e}, \mathbf{v})$ whose joint distribution is $2^{-\Omega(n)}$-statistically close to $(\mathsf{Ber}_{2^{-k}}^n, \mathcal{U}(\{0,1\}^{kn}))$. Moreover,* $\mathsf{Samp}$ *can be implemented in* $\mathrm{NC}^0$.

The sampling method of [AIK08] is as follows: each bit of the Bernoulli output vector $\mathbf{e}$ is obtained by taking the product of $t$ uniformly random input bits; and then to reduce entropy waste, they also apply a *randomness extractor* to the input bits to obtain the pseudorandom part of the output, $\mathbf{v}$. The Bernoulli output is just a product of $t$ and thus computable in $\mathrm{NC}^0$, and [AIK08] constructs an extractor that is also in $\mathrm{NC}^0$ so that the whole sampler is in $\mathrm{NC}^0$.

We now generalize their technique to any real $\tau \in (0, \frac{1}{2}]$. The generalized construction is no longer in $\mathrm{NC}^0$, but has logarithmic depth, which is what we require. For security parameter $\kappa$, let $\tau'$ be the approximation of $\tau$ to $\kappa$ binary places. Note that to a PPT adversary, $\mathsf{Ber}_\tau \overset{c}{\approx} \mathsf{Ber}_{\tau'}$. Now, we can sample from $\mathsf{Ber}_{\tau'}$ by choosing a random value $\mathbf{t} \in \{0,1\}^\kappa$ and outputting 1 if $\mathbf{t} \leq \tau' \cdot 2^\kappa$, and 0 otherwise (for the purposes of the comparison, $\mathbf{t}$ should be interpreted as an integer expressed in binary). Sampling $n$ Bernoulli-distributed bits in this way yields an output distribution which is computationally indistinguishable from the desired $\mathsf{Ber}_\tau^n$ (and this can be done in logarithmic depth). The randomness extractor of [AIK08] is then applied to the input bits just as in their original construction.

A very similar construction can be used to obtain a PRG based on ring-LPN. For easy future reference, we let LPN-PRG denote the above-described LPN-based PRG, and let ring-LPN-PRG denote the ring-LPN based variant PRG.

In our later constructions, we require generation of Bernoulli samples separately from the PRG construction, and derived from pseudorandom – rather than truly random – input. To capture this, we define the following.

**Definition 4.3.** *For security parameter $\kappa$, let $\mathsf{BerSamp}_{n,\tau}^G : \{0,1\}^n \to \{0,1\}^n$ be the function that: takes as input $\mathbf{x} \in \{0,1\}^n$ (which is truly random), applies the pseudorandom generator $G$ to $\mathbf{x}$ to obtain a pseudorandom value $\mathbf{x}' \in \{0,1\}^{(k+k/c)n}$ where $k, c$ are the constants promised in Lemma 4.2 and $\kappa \leq k$, then applies the generalized Bernoulli sampling algorithm of to $\mathbf{x}'$ to obtain a pair $(\mathbf{e}, \mathbf{v})$ whose distribution is computationally indistinguishable from $(\mathsf{Ber}_{2^{-k}}^n, \mathcal{U}(\{0,1\}^{kn}))$, and finally outputs $\mathbf{e}$.*

Note that in the ring-LPN context, we interpret the output of BerSamp as a ring element $e$.

### 4.3 Efficient look-ups using the PRG

We would like to generate a series of pseudorandom values $r_1, \ldots, r_t \in \mathbb{Z}_2^n$ using ring-LPN-PRG, such that each $r_i$ can be looked up in logarithmic time. Let $G_n$ denote the PRG based on ring-LPN-PRG, that maps $n$ bits to $3n$ bits. Let $G_n^{(j)}$ denote the first, middle, or last $n$ bits of the output of PRG3 (for $j = 1, 2, 3$ respectively). We generate $r_1, \ldots, r_t$ in a tree structure as follows.

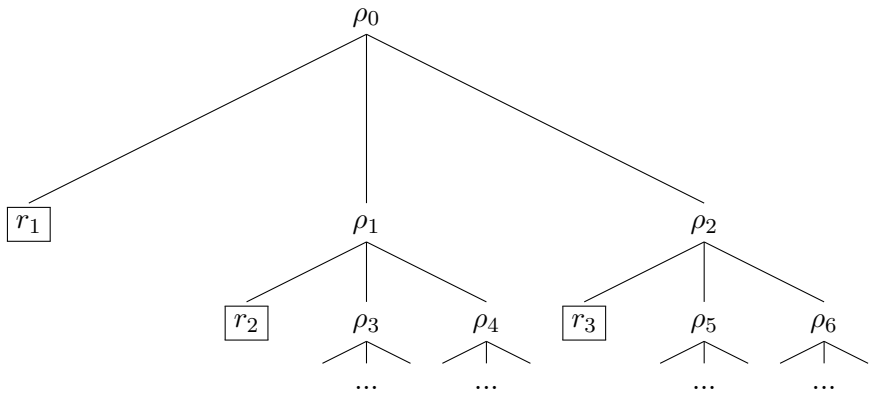


Figure 1: Tree illustrating efficient look-up of pseudorandom values

In Figure 1, $\rho_0 \in \mathbb{Z}_2^n$ is the original (random) input to the PRG. The $\rho_i \in \mathbb{Z}_2^n$ are values which are subsequently pseudorandomly generated, which are used again as input to the PRG to produce

more pseudorandom values: in particular, if $\rho_i$ is a child of $\rho_j$ in the tree, then $\rho_i = G_n^{(2)}(\rho_j)$ if $i$ is even, and $\rho_i = G_n^{(3)}(\rho_j)$ if $i$ is odd. The boxed nodes $r_i \in \mathbb{Z}_2^n$ represent the output pseudorandom values which we want to look up, and they are generated by $r_i = G_n^{(1)}(\rho_j)$ where $\rho_j$ is the parent node of $r_i$. We refer to the boxed nodes as *output nodes* and we call the other nodes *internal nodes*.

For a PRG $G_n : \mathbb{Z}_2^n \to \mathbb{Z}_2^{3n}$, let $\mathsf{lookup}_{G_n}(\rho_0, i) \in \mathbb{Z}_2^n$ denote the $i^{th}$ output value $r_i$ obtained using the above tree method. It is clear that for any polynomial size $i$, the number of PRG evaluations required to look up $r_i$ is logarithmic.

**Lemma 4.4.** *For any PRG $G_n$ and random $\rho_0 \xleftarrow{\$} \mathbb{Z}_2^n$, it holds that*

$$(\mathsf{lookup}_{G_n}(\rho_0, i_1), \ldots, \mathsf{lookup}_{G_n}(\rho_0, i_m)) \overset{c}{\approx} (r_1, \ldots, r_m),$$

*where $r_i \xleftarrow{\$} \mathbb{Z}_2^{3n}$ are uniformly random .*

*Proof.* Since $G_n$ is a PRG, the output of $G_n$ on a random input $\rho$ is indistinguishable from random (as long as the input $\rho$ is not known). In the tree structure, the value at each (non-root) node is obtained by applying $G_n$ to the value at its parent. Therefore, if the value at the root node is random, then it follows (by induction on the depth of the node $N$ in the tree) that the value at any node $N$ is indistinguishable from random as long as none of the values at its ancestors is known.

For any node $N$, the values at its three child nodes are disjoint parts of the output of the PRG (on the value at $N$). Hence, the children are (together) indistinguishable from random as long as the parent is not known. Furthermore, the values at any subset $S$ of nodes of the tree are indistinguishable from random – even if the nodes share parents or ancestors – as long as no ancestor of a node in $S$ is known.

By the design of the tree, no output node is an ancestor of any other output node. Moreover, since the $i_j$ are all distinct, no output value $r_{i_j}$ corresponds to the same output node as another output value $r_{i_{j'}}$. It follows that the outputs $(\mathsf{lookup}_{G_n}(\rho_0, i_1), \ldots, \mathsf{lookup}_{G_n}(\rho_0, i_m))$ are computationally indistinguishable from uniformly random. □

## 4.4 Authentication using the PRG

The approach we take in this section is to replace the Bernoulli noise in a traditional LPN or ring-LPN sample by pseudorandom noise from an LPN-based PRG, and modify the verifier's accept/reject decision to check whether the correct pseudorandom noise was used (instead of a threshold check as in the original protocol). In contrast to the original protocol, the new one has perfect completeness and its communication complexity is unaffected as $\tau \to 1/2$ (although the PRG will be come less efficient). Protocol 3 is an instantiation of this approach based on the simplest LPN-based protocol, HB.

Note that the teal color in the protocol diagram indicates (updating of) the prover's state.

**Lemma 4.5.** *Protocol 3 is perfectly complete.*

*Proof.* This is clear since $\mathsf{lookup}_{G_n}$ is a deterministic algorithm, so by definition, for any honest prover, the verification check must accept. □

**Theorem 4.6.** *Assuming $\mathsf{RingLPN}_{n,\tau}^R$ is hard, Protocol 3 is secure against man-in-the-middle attacks.*

*Proof.* In the following, let $e_j$ denote the noise string for index $j$. Assuming $\mathsf{RingLPN}_{n,\tau}^R$ is hard, $G_n$ is a PRG. Then by Lemma 4.4, the $e_j$ are indistinguishable from random (even given many samples), since the counter $j$ is incremented upon each protocol execution and is thus different between any pair of executions that the adversary can see. Hence $z = a \cdot s + e_j$ is indistinguishable from a ring-LPN sample $z' = a \cdot s + e'$ where $e' \leftarrow \mathsf{Ber}_{1/2}^R$. This, in turn, is (information-theoretically)

**Public parameters.** Security parameter $\kappa \in \mathbb{Z}$, $n \in \mathbb{Z}$ with $n = \mathsf{poly}(\kappa)$, noise rate $\tau \in (0, \frac{1}{2}]$, polynomial ring $R = \mathbb{F}_2[X]/(g)$ with irreducible $g$ of degree $n$.

**Key generation.** $\mathsf{Gen}(1^\kappa)$ samples $s \xleftarrow{\$} R, \mathbf{s}' \xleftarrow{\$} \mathbb{Z}_2^n$ and outputs secret key $(s, \mathbf{s}')$.

**Initial state.** The prover's state consists of $i \in \mathbb{Z}$ initialized to 1.

| $\mathcal{P}(s, \mathbf{s}'; i)$ | | $\mathcal{V}(s, \mathbf{s}')$ |
|---|---|---|
| | $\xleftarrow{a}$ | $a \xleftarrow{\$} R$ |
| $e := \mathsf{lookup}_{G_n}(\mathbf{s}', i)$<br>$z := a \cdot s + e$<br>$i := i + 1$ | $\xrightarrow{z,\ i}$ | accept iff $z + a \cdot s = \mathsf{lookup}_{G_n}(\mathbf{s}', i)$ |

Protocol 3: Enhanced HB protocol based on PRG

indistinguishable from random, regardless of how $a$ is chosen. Thus, we can replace all the $e_j$ by uniformly random strings and the adversary's advantage will change by only a negligible amount.

We now show that it is possible to build a new adversary $\mathcal{A}'$ that only talks to the prover and achieves essentially the same advantage. First, we will replace the honest verifier by a fake verifier $\mathcal{V}'$ who has no access to $s$ or the $e_j$ but still gives essentially the same answers as $\mathcal{V}$. Then we argue that for any man-in-the-middle attack, there is an equally successful *active* attack, and finally prove the active security of the protocol.

$\mathcal{V}'$ works as follows: when queried by $\mathcal{A}$, it chooses a random challenge $a$ (just like $\mathcal{V}$ does). When $\mathcal{A}$ returns an answer $z, j$ (i.e. the second protocol message), there are two cases to consider:

1. $\mathcal{A}$ previously received answer $z', j$ from $\mathcal{P}$, where $z' = a' \cdot s + e_j$ and $a'$ is $\mathcal{A}$'s query to $\mathcal{P}$. Here we have two possibilities:
   (a) $a = a'$ (which could be the case if $\mathcal{A}$ queried $\mathcal{P}$ during the current protocol execution): in this case, if $z = z'$, $\mathcal{V}'$ accepts; else, it rejects.
   (b) $a \neq a'$: $\mathcal{V}'$ always rejects.
2. $\mathcal{A}$ never previously received an answer of the form $z', j$ from $\mathcal{P}$. In this case $\mathcal{V}'$ always rejects.

Consider the first time $\mathcal{A}$ queries the verifier. The challenge produced by $\mathcal{V}$ has exactly the same distribution as the one $\mathcal{V}'$ outputs. Now, in case 1a, notice that $\mathcal{V}$ will accept if and only if $z$ has the correct value $a \cdot s + e_j$: so $\mathcal{V}'$ always makes the same decision as $\mathcal{V}$. In case 1b, $\mathcal{V}$ rejects except with negligible probability, so $\mathcal{V}'$ is statistically close to the right behavior. This is because accepting would imply that $z = a \cdot s + e_j$, but we also have $z' = a' \cdot s + e_j$ so then $s = (z - z')(a - a')^{-1}$. This happens with negligible probability because $s$ is random and $z, z', a, a'$ are all independent of $s$. This holds because all of $\mathcal{P}$'s responses (including $z'$) are independent of $s$. Moreover, since this is the first query, $\mathcal{V}$ has not even looked at $s$ yet, so $a, a'$ and $z$ must be independent of $s$ too. Finally, in case 2, note that no one sees $e_j$ before the adversary produces $z, j$. If $\mathcal{V}$ accepts, we have $z = a \cdot s + e_j$, so $e_j = z - a \cdot s$, which happens with negligible probability since $z, a$ and $s$ are independent of $e_j$.

Therefore, we can modify $\mathcal{V}$ so that it behaves like $\mathcal{V}'$ for the first query, and the adversary's advantage changes at most negligibly as a result. Repeating the same argument for all the queries, we reach the game where $\mathcal{V}$ is entirely replaced by $\mathcal{V}'$, and the adversary's advantage is still at most negligibly different from in the original game.

Since $\mathcal{V}'$ does not possess any secret information, an adversary can run $\mathcal{V}'$ "in his head". So for any adversary $\mathcal{A}$ which has non-negligible advantage in a man-in-the-middle attack, we can

construct an adversary $\mathcal{A}'$ that emulates both $\mathcal{A}$ and $\mathcal{V}'$ "in his head" and achieves the same advantage, but conducting an *active* attack (since he need not interact with the real verifier $\mathcal{V}$).

We now prove the security of the protocol against active attacks. First, note that the prover's message $z$ is indistinguishable from random to any active adversary (even after observing polynomially many honest transcripts), since we established at the beginning of this proof that correctly-formed $z = a \cdot s + e_j$ are indistinguishable from random regardless of the adversary's choice of $a$. So, it remains only to consider the interaction of $\mathcal{A}$ with the verifier. Given a challenge $a$ from an honest verifier $\mathcal{V}$, we argue that $\mathcal{A}$ can have no more than negligible advantage at guessing the (unique) value of $z$ that will cause $\mathcal{V}$ to accept, by considering the following two cases:

1. $\mathcal{A}$ sends an index $i$ that was not used when talking to the honest prover. In this case, we could give the adversary the $e$ values for this $i$ for free (as it is independent of the what happens for the other indices). Now the adversary's task is equivalent to guessing $a \cdot s$, which he cannot do since he has no information about $s$.

2. $\mathcal{A}$ sends an index $i$ that was previously used in a query to the prover. Let $z, i$ be the response (to $a$) from the honest prover. Say the honest verifier sends $a'$ and let $z', i$ be the adversary's response. If there is a non-negligible probability that $z'$ is accepted, then it follows that $z - z' = (a - a') \cdot s$, and thus $s = (z - z')(a - a')^{-1}$. This happens with negligible probability since all of $a, a', z, z'$ were chosen independently of $s$. □

**On choice of parameters.** Suppose we want $k$-bit security for the protocol. It then seems natural to choose parameters $(n, \tau)$ for the underlying LPN problem such that known attacks will require time about $2^k$ to solve it. This will dictate the size of one of the keys, namely $\mathbf{s}'$. However, for $|s|$, i.e. the size of the other key $s$, we only need $k$ bits. This is because the proof of the above theorem shows that the bad events that will cause the real protocol to misbehave occur with probability at most $2^{-|s|}$. Hence the communication of the protocol is always 2 $k$-bit messages, regardless of the choice of $n$ and $\tau$. So, we are free to choose $(n, \tau)$-values that give us the most efficient PRG, or $\tau$ close to $1/2$ if we want maximally hard LPN.

### 4.5 An alternative perspective on the enhanced HB protocol

Although we have derived our protocol from HB, it can also be understood in a different way: we used ring-LPN over a field, and so the prover's answer $a \cdot s + e$ can be interpreted as an unconditionally secure message authentication code (MAC) on message $a$ with key $(s, e)$. That is, the verifier chooses a random message $a$ and the prover is to produce a valid MAC on $a$.

MACs of this form are well known and it is also known that, although a key for an unconditionally secure MAC can usually be used only once, in this case one can reuse the multiplier $(s)$ provided that $e$ is freshly chosen for each message (see e.g. [BDOZ11]): this is what we do pseudorandomly. However, security of our protocol does not follow from MAC security: the standard security notion for MACs simply requires that an adversary who observes a message and a valid MAC cannot produce a different message and valid MAC. We consider a more complicated game where the adversary interacts with prover and verifier concurrently.

## 5 Efficient pseudorandom functions from (ring-)LPN

In this section we show how to construct pseudorandom functions (PRFs) that can be evaluated in poly-logarithmic depth, based on ring-LPN. Concretely, we will construct a weak pseudorandom function (WPRF) and use the result from [NR99] which says that given a WPRF, a PRF can be constructed. Our PRF is parametrized by $n, \ell \in \mathbb{Z}, \tau \in \mathbb{R}, 0 < \tau \leq \frac{1}{2}$ and a polynomial ring $R = \mathbb{F}_2[X]/(g)$ where $g$ is irreducible and has degree $n$, and it is secure assuming the $\mathsf{RingLPN}^n_\tau$

problem is hard and that at most $\ell$ queries are made to the PRF. For a larger number of queries, this same PRF is still secure, but now under stronger assumptions on ring-LPN.

It is straightforward to modify the construction to yield a very similar PRF construction based on LPN rather than ring-LPN.

### 5.1 Definitions

First, we give the definition of pseudorandom functions.

**Definition 5.1** (Pseudorandom function (PRF) [GGM86])**.** *Let $\mathcal{F}$ be a* function family*, i.e. $\mathcal{F}$ is defined by two algorithms $G$ and $F$; The* key generator *$G$ gets as input $1^\kappa$ where $\kappa$ is the security parameter and outputs a key $K = G(1^\kappa)$ (also sometimes called the index). The* evaluation algorithm *$F$ takes as input $K$ and a string $x \in \{0,1\}^n$ where $n = n(\kappa)$ is the input length. It outputs a string $F_K(x) \in Y$ where $Y = Y(\kappa)$ is the output domain. Let $O_U(\kappa)$ be an oracle that on input $x$ returns $R(x)$ where $R : \{0,1\}^{n(\kappa)} \mapsto Y(\kappa)$ is a random function, and let $O_F(\kappa)$ be an oracle that first computes $K = G(1^\kappa)$ and then on input $x$ returns $F_K(x)$.*

*$\mathcal{F}$ is said to be pseudorandom (or a secure PRF), if for any* PPT *algorithm $\mathcal{A}$ that outputs a single bit, it holds that $|Pr[\mathcal{A}^{O_U(\kappa)} = 1] - \Pr[\mathcal{A}^{O_F(\kappa)} = 1]| \leq \mathsf{negl}(\kappa)$.*

We also need to define weak pseudorandom functions (WPRFs) [NR99], which are basically like PRFs, except that the adversary does not get to choose the inputs to the function: instead, they are always uniformly chosen.

**Definition 5.2** (Weak pseudorandom function (WPRF))**.** *Let $O_U^W(\kappa)$ be an oracle that takes no input but when queried returns $x, R(x)$ where $x$ is chosen uniformly and $R : \{0,1\}^{n(\kappa)} \mapsto Y(\kappa)$ is a random function. Let $O_F^W(\kappa)$ be an oracle that first computes $K = G(1^\kappa)$ and then returns $x, F_K(x)$ where $x$ is chosen uniformly.*

*$\mathcal{F}$ is said to be weak pseudorandom (or a secure WPRF), if for any* PPT *algorithm $\mathcal{A}$ that outputs a single bit, it holds that $|\Pr[\mathcal{A}^{O_U^W(\kappa)} = 1] - \Pr[\mathcal{A}^{O_F^W(\kappa)} = 1]| \leq \mathsf{negl}(\kappa)$.*

[NR99] also introduced the concept of a *synthesizer* (alongside WPRFs). We do not need to formally define synthesizers here; however, we make use of the following result.

**Theorem 5.3** ([NR99])**.** *A secure PRF can be constructed from any synthesizer, and a synthesizer can be constructed from any WPRF whose input and output sets are equal. Furthermore, if the WPRF can be evaluated in poly-logarithmic depth, then so can the resulting PRF.*

In [BPR12] a construction of a synthesizer was given based on the LWE problem, but their approach will not work for us. Instead, our goal here is to construct a WPRF based on (ring-)LPN.

Finally, it is useful to define $\ell$-independent hashing.

**Definition 5.4** ($\ell$-independent hashing)**.** *Let $\mathcal{H}$ be a family of hash functions, where each hash function $h \in \mathcal{H}$ maps from a domain $D$ to a codomain $C$. $\mathcal{H}$ is said to be* $\ell$-independent *if for any fixed sequence of $\ell$ distinct inputs $(k_1, \ldots, k_\ell) \in D^\ell$ and for $h \leftarrow \mathcal{H}$ chosen at random, the sequence $(h(x_1), \ldots, h(x_\ell))$ is uniformly random in $C^\ell$.*

### 5.2 The PRF construction

We construct a WPRF family $\mathcal{F}^{\mathsf{RingLPN}}$, parametrized by $n, \ell \in \mathbb{Z}, \tau \in (0, \frac{1}{2}]$ and $R = \mathbb{F}_2[X]/(g)$ a polynomial ring with irreducible $g$ of degree $n$. The input to our WPRF will be a random ring element $a \in R$, and the key will be a random $s \in R$. Let $G$ be a PRG based on ring-LPN-PRG. We interpret the output of $G$ on input $s$ as follows. $G(s)$ produces two pseudorandom outputs:

- a $((\ell + 1) \cdot n)$-bit string $\mathbf{x}_s$, and

- a description of a hash function $h_s$, chosen from a family of $2\ell$-wise independent hash functions that map $(\ell+1)\cdot n$ bits to $n$ bits.

Now we define $\mathcal{F}^{\mathsf{RingLPN}} = \{F_s\}$ (parametrized by $n, \ell, \tau, R$) as follows:

$$F_s(a) = a \cdot s + \mathsf{BerSamp}_\tau^n(h_s(\mathbf{x}_s + a)).$$

Note that the sum $\mathbf{x}_s + a$ above is the bitwise exclusive-or of $\mathbf{x}$ with $a$ interpreted as a bit vector, including padding (with zeros) so that the vector lengths are equal.

**Remark.** A hash function satisfying the above properties can be chosen as a random polynomial of degree at most $2\ell$ over the field with $2^{(\ell+1)\cdot n}$ elements. To compute the hash function, evaluate the polynomial at the input point, and output the least significant $n$ bits of the result.

**Lemma 5.5.** *The functions in $\mathcal{F}^{\mathsf{RingLPN}}$ can be evaluated in depth poly-logarithmic in $n$ and $\ell$.*

*Proof.* First, note that the product of polynomials $a \cdot s$ can be done in depth $O(\log(n))$, for the first part of the output of $F_s$. Now we consider the evaluation of the PRG.

Let $G_{2\times} : \{0,1\}^n \to \{0,1\}^{2n}$ denote a pseudorandom generator based on ring-LPN-PRG where the output length is double the input length, and let $G^0_{2\times}(x), G^1_{2\times}(x)$ denote the first and last $n$ bits of the output of $G_{2\times}(x)$, respectively. Since our required output size for the PRG, $|\mathbf{x}_s| + |h_s|$, is polynomial in $\ell$ and $n$, we can apply the PRG $G_{2\times}$ recursively to the two halves of its output $G^0_{2\times}, G^1_{2\times}$ to obtain the required number of output bits in depth logarithmic in the output length. In total, this calls for poly-logarithmic depth of computation, since each PRG evaluation requires logarithmic depth.

Finally, the hash function is computed by evaluating a polynomial of degree at most $2\ell$ over the field with $2^{(\ell+1)\cdot n}$ elements. It is well known that this can be done in depth that is logarithmic in both degree and bit size of field elements. The lemma follows. □

Before showing our initial security result, we need the following technical lemma. It can be seen as a generalization of the left-over hash lemma and has a similar proof.

**Lemma 5.6** ([DFMV13])**.** *Let $(X_1, X_2, \ldots, X_\ell) \in \mathcal{X}^\ell$ be $\ell$ (possibly dependent) random variables such that $H_\infty(X_i) \geq \beta$ and $(X_1, \ldots, X_\ell)$ are pairwise different. Let $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$ be a family of $2\ell$-wise independent hash functions, with $|\mathcal{Y}| = 2^k$. Then for random $h \leftarrow \mathcal{H}$ we have that the statistical distance satisfies*

$$\Delta((h, h(X_1), h(X_2), \ldots, h(X_\ell)); (h, U^1_{\mathcal{Y}}, \ldots, U^\ell_{\mathcal{Y}})) \leq \frac{\ell}{2} \cdot 2^{(\ell \cdot k - \beta)/2},$$

*where $U^1_{\mathcal{Y}}, \ldots, U^\ell_{\mathcal{Y}}$ are $\ell$ independent and uniformly distributed variables.*

In the theorem below, we prove that this lemma implies that our construction is a secure WPRF if the adversary makes at most $\ell$ queries, and after this we consider what happens for more than $\ell$ queries[1].

**Theorem 5.7.** *Under the $\mathsf{RingLPN}^R_{n,\tau}$ assumption, $\mathcal{F}^{\mathsf{RingLPN}}$ is a weak pseudorandom function family for any adversary making no more than $\ell$ queries to its oracle.*

*Proof.* By the assumption, $G$ is a secure pseudorandom generator. We can therefore modify the oracle $O^W_F$ from Definition 5.2 so that it uses uniformly chosen $x_S$ and $h_S$, and the advantage of any adversary will change only by a negligible amount.

[1] If we were *only* interested in security for $\ell$ queries this could be done very easily, even without relying on LPN: we could use an $\ell$-wise independent hash function $h$ as secret key and output $h(x)$ on input $x$. However, such construction is completely insecure if more than $\ell$ queries are asked.

Let $a_1, \ldots, a_\ell$ be the (uniformly random) inputs occurring in the oracle responses. Let $U_{(\ell+1)\cdot n}$ be a random variable that is uniformly distributed over the set of $((\ell+1)\cdot n)$-bit strings. Consider the random variables $U_{(\ell+1)\cdot n} + a_1, U_{(\ell+1)\cdot n} + a_2, \ldots, U_{(\ell+1)\cdot n} + a_\ell$. Each variable $U_{(\ell+1)\cdot n} + a_i$ has entropy $H(U_{(\ell+1)\cdot n} + a_i) = H_\infty(U_{(\ell+1)\cdot n} + a_i) = (\ell+1)\cdot n$. They are not independent, but they are pairwise different because the $a_i$'s are distinct (with overwhelming probability). Therefore, by Lemma 5.6,

$$\{h_S, h_S(U_{(\ell+1)\cdot n} + a_1), \ldots, h_S(U_{(\ell+1)\cdot n} + a_\ell)\} \stackrel{s}{\approx} \{h_S, U^1_{\lceil n\cdot H(\tau)\rfloor}, \ldots U^\ell_{\lceil n\cdot H(\tau)\rfloor}\},$$

i.e. in the lemma we have $k = n$ and $\beta = (\ell+1)\cdot n$, so the distance is $\frac{\ell}{2}2^{-n}$ which is negligible if $\ell = \mathsf{poly}(n)$. We can therefore modify oracle $O_F^W$ again so it outputs $(a_i, a_i \cdot s + \mathsf{BerSamp}^n_\tau(\mathbf{e}_i))$ where $e_i$ is a fresh random value chosen for each query. This is in turn indistinguishable from $(a_i, a_i \cdot s + e'_i)$ where $e'_i \leftarrow \mathsf{Ber}^{R,n}_\tau$. Finally, we can replace $a_i \cdot s + e'_i$ by a uniformly random value under the $\mathsf{RingLPN}^R_{n,\tau}$ assumption, so the final modified $O_F^W$ is computationally indistinguishable from $O_R^W$. □

It follows from the construction in [NR99] that if we build a PRF from a WPRF and the adversary makes no more than $\ell$ queries to the PRF, then no more than $\ell$ queries are made to any instance of the WPRF (the construction uses several instances of the WPRF with different keys). Hence, if the application of the PRF can guarantee some (polynomial) upper bound on $\ell$, Lemma 5.7 is already sufficient, and we need only assume standard hardness of LPN. For example, this will be the case for our PRF-based authentication protocol given in Section 6. Note, also, that $\ell$ can be chosen freely without affecting the required secret key size.

In some settings, such a bound on the number of queries is not a realistic assumption, but we can still prove security of the WPRF in general, under a stronger assumption on the ring-LPN problem: namely, that even if the noise elements $e$ in the samples $(a, a \cdot s + e)$ are only $\ell$-wise independent between samples (but still each distributed according to $\mathsf{Ber}^R_\tau$), then as long as $\ell$ is large enough, it is still hard to distinguish any polynomial number of samples from uniformly random. Formally, we make the following conjecture.

**Definition 5.8.** *Let $\mathcal{D}^{n,\tau}_{\ell,R}(\mathbf{a}_1, \ldots, \mathbf{a}_k)$ be the distribution defined by the following sampling procedure: choose a random $\mathbf{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_2^{(\ell+1)\cdot n}$ and a $2\ell$-wise independent hash function $h$ with input size $(\ell+1)\cdot n$ bits and output size $n$ bits (a polynomial, as described above), and then output*

$$\mathsf{BerSamp}^n_\tau(h(\mathbf{u} + \mathbf{a}_1)), \ldots, \mathsf{BerSamp}^n_\tau((\mathbf{u} + \mathbf{a}_k))$$

*(with zero-padding where necessary to match summand lengths).*

**Conjecture 5.9.** *Let $\kappa$ be a security parameter, let $n, k = \mathsf{poly}(\kappa)$, $\tau \in (0, \frac{1}{2}]$, and $R = \mathbb{F}_2[X]/(g)$ with irreducible $g$ of degree $n$. Let $a_1, \ldots, a_k \stackrel{\$}{\leftarrow} R$ and $s \stackrel{\$}{\leftarrow} R$, and $e_1, \ldots, e_k \leftarrow \mathcal{D}^{n,\tau}_{\ell,R}(a_1, \ldots, a_k)$. Then there exists a polynomial $f$ such that if we set $\ell = f(\kappa)$, then it holds that*

$$((a_1, a_1 \cdot s + e_1), \ldots, (a_k, a_k \cdot s + e_k)) \stackrel{c}{\approx} ((a_1, u_1), \ldots, (a_k, u_k))$$

*where the $u_i \stackrel{\$}{\leftarrow} R$ are uniformly random.*

Of course, this assumption is new and should be studied carefully before it is used. As arguments in favor we can note that for any polynomial value of $k$, the probability that the $a_i$'s are not distinct is negligible, and therefore by Lemma 5.6 the $e_i$'s are $\ell$-wise independent. Hence any attack that is faster than trying to solve ring-LPN directly must consider at least $\ell + 1$ of the $k$ instances simultaneously. Furthermore, the $\ell$-wise independence holds even if the hash function $h$ is given. In our setting, the adversary does not get $h$ so this should add to the difficulty of the problem.

We can now show the security of the PRF in general:

**Theorem 5.10.** *Assuming Conjecture 5.9, $\mathcal{F}^{\mathsf{RingLPN}}$ is a secure WPRF.*

*Proof.* The conjecture clearly implies that ring-LPN is hard for the corresponding parameters, so the generator $G$ used in the construction of $\mathcal{F}^{\mathsf{RingLPN}}$ is a secure PRG. Thus, we can modify the oracle $O_F$ as in the proof of Theorem 5.7 so that the $\mathbf{x}_s$ and $h_s$ are uniformly chosen, and the adversary's advantage increases by at most a negligible amount as a result of this modification.

Let $a_1, \ldots, a_k$ be the inputs occurring in the oracle responses. Then the distribution of the "noise components" $\mathsf{BerSamp}_\tau^n(h_s(\mathbf{x}_s + a_i))$ as defined in the construction of $\mathcal{F}^{\mathsf{RingLPN}}$ is exactly according to $\mathcal{D}_{\ell,R}^{n,\tau}(a_1, \ldots, a_k)$. Hence, the conjecture implies the outputs are indistinguishable from random. □

As before, by the construction of [NR99], $\mathcal{F}^{\mathsf{RingLPN}}$ can be used to build a secure PRF family (again, assuming that Conjecture 5.9 holds), as summarized in the following theorem.

**Theorem 5.11.** *Let $\widetilde{\mathcal{F}}^{\mathsf{RingLPN}} = \{\widetilde{F}_s\}$ be the PRF family obtained by applying the [NR99] technique to the WPRF family $\mathcal{F}^{\mathsf{RingLPN}}$, for parameters $n, \ell \in \mathbb{Z}, \tau \in (0, \frac{1}{2}], R = \mathbb{F}_2[X]/(g)$. Then, if $\mathsf{RingLPN}_{n,\tau}^R$ is hard, $\widetilde{\mathcal{F}}^{\mathsf{RingLPN}}$ is a secure PRF family for any adversary making at most $\ell$ queries. Moreover, if Conjecture 5.9 holds, then $\widetilde{\mathcal{F}}^{\mathsf{RingLPN}}$ is a secure PRF family (for any* PPT *adversary).*

*Proof.* Follows directly from Theorems 5.3, 5.7, and 5.10. □

**Remark.** We can straightforwardly modify the above construction to be based on LPN rather than ring-LPN. For best efficiency, the construction based on LPN should take matrices $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ as input, and operate on $l$ secrets $\mathbf{s}_i \in \mathbb{Z}_2^n$ in parallel, for some $m, l = \mathsf{poly}(n)$. More concretely, this yields the function family $\mathcal{F}^{\mathsf{LPN}} = \{F_{\mathbf{S}}\}$ (parametrized by $n, \ell, m, l$) defined by:

$$F_{\mathbf{S}}(\mathbf{A}) = \mathbf{AS} + \mathsf{BerSamp}_\tau^{m \times l}(h_{\mathbf{S}}(\mathbf{x}_{\mathbf{S}} + \mathbf{A})),$$

where $\mathbf{S}$ is a matrix whose columns are the secret vectors $\mathbf{s}_i$. Note that the re-use of a single public matrix $\mathbf{A}$ for multiple secrets $\mathbf{s}_i$ is known to be secure under the LPN assumption.

# 6 Authentication using pseudorandom functions

In this section we present an authentication protocol based on the secure PRF construction from Section 5. $\ell$ denotes the maximum number of queries the adversary can make in order to guarantee security under the ring-LPN assumption (according to Theorem 5.7). Note that $G_n$ denotes the PRG based on ring-LPN-PRG, that maps $n$ bits to $3n$ bits (as defined in Section 4).

**Lemma 6.1.** *Protocol 4 is perfectly complete.*

*Proof.* This is clear since $\mathsf{lookup}_{G_n}$ is a deterministic algorithm, so by definition, for any honest prover, the verification check must accept. □

**Theorem 6.2.** *Assuming $\mathsf{RingLPN}_{n,\tau}^R$ is hard, Protocol 4 is secure against man-in-the-middle attacks.*

*Proof.* Assuming $\mathsf{RingLPN}_{n,\tau}^R$ is hard, $G_n$ is a PRG. Then by Lemma 4.4, since the secret $s_0$ is chosen at random, the values $s = \mathsf{lookup}_{G_n}(s_0, i)$ for $i = 1, 2, \ldots$ used as PRF seeds in the protocol are indistinguishable from random. By Theorem 5.11, $\widetilde{\mathcal{F}}^{\mathsf{RingLPN}}$ is a secure PRF family since we assume that the adversary makes no more than $\ell$ queries. It follows that $\widetilde{F}_s$ for a random seed $s$ is indistinguishable from a random oracle (for an adversary $\mathcal{A}$ who is given oracle access to $\widetilde{F}_s$). Hence, interacting with the honest prover $\mathcal{P}$ will give the adversary outputs $z$ which are indistinguishable from that of a random oracle, and therefore the adversary can gain no more than negligible advantage from interacting with $\mathcal{P}$.

**Public parameters.** Security parameter $\kappa \in \mathbb{Z}$, $n, \ell \in \mathbb{Z}$ with $n, \ell = \mathsf{poly}(\kappa)$, noise rate $\tau \in (0, \frac{1}{2}]$, polynomial ring $R = \mathbb{F}_2[X]/(g)$ with irreducible $g$ of degree $n$.

**Key generation.** $\mathsf{Gen}(1^\kappa)$ samples secret key $s_0 \xleftarrow{\$} R$.

**Initial state.** The prover's state consists of $i, j \in \mathbb{Z}$, both initialized to 1.

| $\mathcal{P}(s_0; i, j)$ | | $\mathcal{V}(s_0)$ |
|---|---|---|
| | $\xleftarrow{a}$ | $a \xleftarrow{\$} R$ |
| $s := \mathsf{lookup}_{G_n}(s_0, i)$ | | |
| $z := \widetilde{F}_s(a)$ | | |
| $j := j + 1$ | | $s := \mathsf{lookup}_{G_n}(s_0, i)$ |
| if $j = \ell + 1$ then $\{i := i + 1; j = 1\}$ | $\xrightarrow{z,\ i}$ | accept iff $z = \widetilde{F}_s(a)$ |

Protocol 4: Authentication using pseudorandom functions

The other option available to the adversary is to interact with the honest verifier $\mathcal{V}$ and observe his accept/reject decisions. Clearly, the adversary gains no information from the initial message $a$ since it is just a random ring element. Now we consider whether the adversary can gain an advantage from access to the accept/reject decisions. For initial message $a$, the verifier $\mathcal{V}$ accepts if and only if $z = \widetilde{F}_s(a)$. $\mathcal{V}$'s response for any given input $a$ is distributed as one of these two cases:

1. If $\mathcal{A}$ already queried the prover $\mathcal{P}$ on input $a$, then $\mathcal{A}$ already knows (given $a$) the true distribution of $z$, so the verifier's response gives the adversary no additional information.

2. Otherwise, if $\mathcal{A}$ has not queried $\mathcal{P}$ on input $a$, then since $\mathcal{F}^{\mathsf{RingLPN}}$ is a PRF family (for up to $\ell$ queries) and $s$ is randomly chosen and unknown to $\mathcal{A}$, the verifier's decision on any transcript $(a, z')$ where $z'$ is chosen by $\mathcal{A}$ is indistinguishable, to the adversary, from the decision procedure that always outputs reject. So, again, the verifier's response gives the adversary no additional information.

Let $D_1$ denote the distribution defined in item 1, and $D_2$ denote the distribution defined in item 2 above. Now consider an adversary who makes polynomially many queries $a_1, \ldots, a_k$. In this case, by a hybrid argument replacing each $a_i$ by $b_i$ for each $i$ one by one, we argue that the joint distribution of the verifier's responses to all the queries is indistinguishable from the joint distribution of $b_1, \ldots, b_k$ where each $b_i$ is drawn (independently) from either $D_1$ or $D_2$. It follows that the adversary who makes polynomially many queries gains at most negligible advantage from interacting with the honest verifier. Therefore, the protocol is secure against MIM attacks. □

**Theorem 6.3.** *Assuming Conjecture 5.9 holds, Protocol 4 is secure against man-in-the-middle attacks even for adversaries with the power to reset the prover during the query stage.*

*Proof.* The conjecture clearly implies that ring-LPN is hard for the corresponding parameters, so the generator $G_n$ is a secure PRG. Moreover, by Theorem 5.11, $\widetilde{\mathcal{F}}^{\mathsf{RingLPN}}$ is a secure PRF family. The rest of the proof is exactly as in the proof of Theorem 6.2. □

In fact, if Conjecture 5.9 holds, then a straightforward modification to Protocol 4 (essentially, removing the counters $i, j$) gives a simpler and *stateless* authentication protocol.

# 7 A generalized PRF construction

In this section we show a PRF construction that is based on functions with certain properties rather than a specific problem such as LPN or ring-LPN. Identifying these essential properties then opens a way to base the construction on other problems. Finally, following this paradigm, we show that the subset sum problem can also be used as a basis for a PRF.

Suppose that we have a family of functions $\mathcal{G} = \{g_{\mathbf{r}}\}$ with the index $\mathbf{r} \in \{0,1\}^n$ and

$$g_{\mathbf{r}} : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^m,$$

where $n = n(\kappa), m = m(\kappa), d = d(\kappa)$ are functions of a security parameter $\kappa$. Furthermore, suppose that for some distributions $\mathcal{R}, \mathcal{X}$ over $\{0,1\}^n, \{0,1\}^d$ respectively, it holds that: for $\mathbf{r} \leftarrow \mathcal{R}$, (polynomially many) samples of the form $(\mathbf{a}, g_{\mathbf{r}}(\mathbf{a}, \mathbf{x}))$ for $\mathbf{a} \xleftarrow{\$} \{0,1\}^m, \mathbf{x} \leftarrow \mathcal{X}$ (with $\mathbf{r}$ unknown and fixed across all samples) are computationally indistinguishable from samples $(\mathbf{a}, \mathbf{u})$ where $\mathbf{u}$ is chosen uniformly at random.

In the case of LPN, for example, $\mathbf{r}$ corresponds to the secret $\mathbf{s}$, $\mathbf{a}$ corresponds to the public randomness $\mathbf{a}$, and $\mathbf{x}$ corresponds to the noise $\mathbf{e}$. An advantage of this abstraction is that the well-known subset sum problem can also be phrased in this way, and thus we can construct PRFs based on the hardness of subset sum.

## 7.1 Efficient pseudorandom functions from subset sum

**Notation.** In this section, $+$ denotes integer addition (in a field, where applicable) and $\oplus$ denotes (component-wise) binary addition, i.e. exclusive-or.

### 7.1.1 Subset sum

The subset sum problem is defined by parameters $d$ and $M$ as follows: given $d$ numbers $a_1, \ldots, a_d \in \mathbb{Z}_M$, and a target number $t \in \mathbb{Z}_M$, find a subset $S \subset \{1, \ldots, d\}$ such that $\sum_{i \in S} a_i = t \mod M$. We consider a function $g$ defined as follows:

$$g(\mathbf{a}, S) = \sum_{i \in S} \mathbf{a}_i \mod M.$$

for $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_d)$ sampled uniformly at random in $\mathbb{Z}_M^d$. The subset sum problem can then be viewed as that of inverting $g$.

We adopt the notation from [LPS10] for describing the subset sum function: we identify the subset $S \subset \{1, \ldots, d\}$ with its incidence vector $\mathbf{s} = (\mathbf{s}_0, \ldots, \mathbf{s}_{d-1}) \in \{0,1\}^d$. Given numbers in $\mathbb{Z}_M$ for $M = q^n$ we write the matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_q^{n \times d}$ and define the subset sum product, $\mathbf{A} \odot \mathbf{s} = \mathbf{t} \in \mathbb{Z}_q^n$. Concretely $\mathbf{A} \odot \mathbf{s} = \mathbf{A}\mathbf{s} + \mathbf{c}$, where $\mathbf{c} \in \mathbb{Z}_q^n$ is the vector representing the carries that arise when doing addition of numbers. Here we interpret the columns of $\mathbf{A}$ as numbers in $\mathbb{Z}_q^n$ written in base $q$ and sum all the elements from the columns where $s_i = 1$. In other words we have

$$\sum_{i=0}^{n-1} \mathbf{t}_i \mathbf{q}_i = \left( \sum_{j=0}^{d-1} \mathbf{s}_j \sum_{i=0}^{n-1} \mathbf{a}_{ij} q^i \right) \mod q^n.$$

As an example consider $M = q^n, q = 10, n = d = 3, \mathbf{a} = (623, 425, 519), \mathbf{S} = \{1, 3\}$. Computing the subset sum we get: $f_{\mathbf{a}}(\mathbf{S}) = 623 + 519 \mod 10^3 = 142$. In the matrix notation we get

$$\mathbf{A} \odot \mathbf{s} = \mathbf{A}\mathbf{s} + \mathbf{c} = \begin{bmatrix} 6 & 4 & 5 \\ 2 & 2 & 1 \\ 3 & 5 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$$

with all operations done modulo $q$.

**Hardness.** According to our current knowledge, the hardness of a concrete instance of the subset sum problem depends on the relation between $d$ and $M$. For this purpose we consider the *density* of a problem instance, $\rho = d/\log(M)$. In [IN96] it is shown that the problem gets harder as the density gets closer to 1. When the density is less than $1/d$ or larger than $d/\log^2(d)$ there exist algorithms for solving the problem in polynomial time [LO85; Fri86; FP05; Lyu05; Sha08]. Thus, for cryptographic constructions based on this problem, concrete choices of parameters have to stay within the intervening range where there are currently no known polynomial time algorithms.

### 7.1.2 PRF from subset sum

Impagliazzo and Naor [IN96] showed that if the subset sum function is hard to invert, i.e. one-way, then it is also a pseudorandom generator:

**Theorem 7.1.** *If the subset sum function $g(\mathbf{a}, S)$ is one-way, then $\{\mathbf{a}, g(\mathbf{a}, S)\}$ and $\{\mathbf{a}, \mathbf{u}\}$ are computationally indistinguishable, where $\mathbf{a} \in \mathbb{Z}_M^d$, $S \subset \{1, \ldots, d\}$, and $\mathbf{u} \in \mathbb{Z}_M$ are chosen independently at random.*

Given this theorem, we can apply the abstraction detailed at the beginning of Appendix **??**, and conclude that the construction of Section 5.2 can also be based upon the subset sum problem. In this case, concretely, the WPRF $\mathcal{F}^{\mathsf{SubsetSum}} = \{F_{\mathbf{S}}\}$ is given by

$$F_{\mathbf{S}} = \mathbf{A} \odot h_{\mathbf{S}}(\mathbf{x}_{\mathbf{S}} \oplus \mathbf{A}),$$

where $\mathbf{S} \leftarrow \{0,1\}^{n \times n}$, $h_{\mathbf{S}}$, and $\mathbf{x}_{\mathbf{S}}$ are generated pseudorandomly by the PRG $g$ established above, and $\mathbf{S}$ represents the "key" of the PRF.

This yields a secure PRF assuming the hardness of the subset sum problem for constant density, for an adversary that makes no more than a certain threshold $\ell$ of queries. Security for general PPT adversaries is based on a stronger assumption on subset sum, where the secret subset $\mathbf{s}'$ in samples $(\mathbf{A}, \mathbf{A} \odot \mathbf{s}')$ is chosen (afresh for each sample) according to an $\ell$-wise independent distribution.

**Remark.** Note that here, the index $\mathbf{r}$ from the abstraction is not relevant, i.e. it can be considered to be drawn from a set containing one element. This is because in the subset sum problem, none of the secret randomness $\mathbf{s}$ can be reused across multiple samples without compromising security. On the other hand, with LPN, the secret $\mathbf{s}$ can be used across many samples, but the secret noise vector $\mathbf{e}$ cannot. $\mathbf{r}$ can be considered to be the secret randomness that is reused across samples.

## 8 Conclusion

We have seen two approaches to improving authentication protocols based on LPN and ring-LPN as the noise rate comes closer to the maximal value $\tau \leftarrow \frac{1}{2}$, that is, as the computational assumption gets weaker. We have also seen a new method to construct a PRF from LPN or other hard problems with a similar structure, which can be computed in low depth and is secure agaist a bounded number of queries. Moreover, for a larger number of queries, this PRF construction has a graceful degradation to a stronger assumption.

## References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. "Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems". In: *Advances in Cryptology - CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 595–618. ISBN: 978-3-642-03355-1. DOI: 10.1007/978-3-642-03356-8_35. URL: http://dx.doi.org/10.1007/978-3-642-03356-8_35.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. "On Pseudorandom Generators with Linear Stretch in $NC^0$". In: *Computational Complexity* 17.1 (2008), pp. 38–69.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. "Pseudorandom Functions and Lattices". In: *EUROCRYPT*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 719–737. ISBN: 978-3-642-29010-7.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. "Semi-homomorphic Encryption and Multiparty Computation". In: *EUROCRYPT*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 169–188. ISBN: 978-3-642-20464-7.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. "Cryptographic Primitives Based on Hard Learning Problems". In: *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '93. London, UK, UK: Springer-Verlag, 1994, pp. 278–291. ISBN: 3-540-57766-1. URL: http://dl.acm.org/citation.cfm?id=646758.759585.

[DFMV13] Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. "Tamper Resilient Cryptography Without Self-Destruct". In: *IACR Cryptology ePrint Archive* 2013 (2013), p. 124.

[DKPW12] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. "Message Authentication, Revisited". In: *EUROCRYPT*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 355–374. ISBN: 978-3-642-29010-7.

[FP05] Abraham Flaxman and Bartosz Przydatek. "Solving Medium-Density Subset Sum Problems in Expected Polynomial Time". In: *STACS*. Ed. by Volker Diekert and Bruno Durand. Vol. 3404. Lecture Notes in Computer Science. Springer, 2005, pp. 305–314. ISBN: 3-540-24998-2.

[Fri86] Alan M. Frieze. "On the Lagarias-Odlyzko Algorithm for the Subset Sum Problem". In: *SIAM J. Comput.* 15.2 (1986), pp. 536–539.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (1986), pp. 792–807.

[Hey+12] Stefan Heyse et al. "Lapin: An Efficient Authentication Protocol Based on Ring-LPN". In: *FSE*. Ed. by Anne Canteaut. Vol. 7549. Lecture Notes in Computer Science. Springer, 2012, pp. 346–365. ISBN: 978-3-642-34046-8.

[HB01] NicholasJ. Hopper and Manuel Blum. "Secure Human Identification Protocols". English. In: *Advances in Cryptology ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 52–66. ISBN: 978-3-540-42987-6. DOI: 10.1007/3-540-45682-1_4. URL: http://dx.doi.org/10.1007/3-540-45682-1_4.

[IN96] Russell Impagliazzo and Moni Naor. "Efficient Cryptographic Schemes Provably as Secure as Subset Sum". In: *J. Cryptology* 9.4 (1996), pp. 199–216.

[JW05] Ari Juels and Stephen A. Weis. "Authenticating Pervasive Devices with Human Protocols". In: *CRYPTO*. Ed. by Victor Shoup. Vol. 3621. Lecture Notes in Computer Science. Springer, 2005, pp. 293–308. ISBN: 3-540-28114-2.

[KSS10] Jonathan Katz, Ji Sun Shin, and Adam Smith. "Parallel and Concurrent Security of the HB and $HB^+$ Protocols". In: *J. Cryptology* 23.3 (2010), pp. 402–421.

[Kil+11] Eike Kiltz et al. "Efficient Authentication from Hard Learning Problems". In: *EUROCRYPT*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 7–26. ISBN: 978-3-642-20464-7.

[LO85] J. C. Lagarias and Andrew M. Odlyzko. “Solving Low-Density Subset Sum Problems”. In: *J. ACM* 32.1 (1985), pp. 229–246.

[Lyu05] Vadim Lyubashevsky. “The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem”. In: *APPROX-RANDOM*. Ed. by Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan. Vol. 3624. Lecture Notes in Computer Science. Springer, 2005, pp. 378–389. ISBN: 3-540-28239-4.

[LPS10] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. “Public-Key Cryptographic Primitives Provably as Secure as Subset Sum”. In: *TCC*. Ed. by Daniele Micciancio. Vol. 5978. Lecture Notes in Computer Science. Springer, 2010, pp. 382–400. ISBN: 978-3-642-11798-5.

[NR99] Moni Naor and Omer Reingold. “Synthesizers and Their Application to the Parallel Construction of Pseudo-Random Functions”. In: *J. Comput. Syst. Sci.* 58.2 (1999), pp. 336–375.

[Pat11] Kenneth G. Paterson, ed. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011. ISBN: 978-3-642-20464-7.

[PJ12] David Pointcheval and Thomas Johansson, eds. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012. ISBN: 978-3-642-29010-7.

[PP03] David Pointcheval and Guillaume Poupard. “A New NP-Complete Problem and Public-Key Identification”. In: *Des. Codes Cryptography* 28.1 (2003), pp. 5–31.

[Sha08] Andrew Shallue. “An Improved Multi-set Algorithm for the Dense Subset Sum Problem”. In: *ANTS*. Ed. by Alfred J. van der Poorten and Andreas Stein. Vol. 5011. Lecture Notes in Computer Science. Springer, 2008, pp. 416–429. ISBN: 978-3-540-79455-4.