

Resource Bounded Randomness for Instantiating Cryptographic Security

Siddharth Namachivayam

Signatures

- Why do we sign messages? What is the social function of a signature?
- Signatures allow us to establish *public knowledge* of who said what.
- But how is public knowledge established?
- A good signature better be *unforgeable*.
- Extremely important for the operation of legal and financial contracts.

Cryptographic Security

- Unfortunately, traditional handwritten signatures can be easily forged.
- Modern cryptography provides a potential solution (e.g. RSA).
- But what guarantees does a signature scheme like RSA actually provide?
- Cryptographers appeal to a variety of different *security notions* to cash out in what sense a particular scheme is “unforgeable.”
- We now formalize what is meant by a signature scheme and present the security notion we will study.

Housekeeping

- $\{0,1\}^*$ denotes the set of finite binary strings.
- $\{0,1\}^\infty$ denotes the set of infinite binary sequences (Cantor space).
- A *variable-hash* is a function $H : \mathbb{N}^+ \times \{0,1\}^* \rightarrow \{0,1\}^*$ such that $\forall (n, x) \in \mathbb{N}^+ \times \{0,1\}^*, |H(n, x)| = n$.
- A *n-hash* is a function $h : \{0,1\}^* \rightarrow \{0,1\}^*$ such that $\forall x \in \{0,1\}^*, |h(x)| = n$.
- $H_n(_)$ will be shorthand for $H(n, _)$.
- $a^{-1} : \{0,1\}^* \rightarrow \mathbb{N}^+$ enumerates strings according to the lexicographic order.
- $b^{-1} : \mathbb{N}^+ \times \mathbb{N}^+ \rightarrow \mathbb{N}^+$ is the bijection $(m, n) \mapsto (m + n - 2)(m + n - 1)/2 + n$.
- $c : \mathbb{N}^+ \rightarrow \mathbb{N}^+ \times \{0,1\}^*$ is the bijection $n \mapsto (b_1(n), a(b_2(n)))$.
- We can establish a bijection between variable-hashes and elements of $\{0,1\}^\infty$ by mapping $H \mapsto H(c(1))H(c(2))H(c(3))\dots$
- We will often identify variable-hashes with elements of $\{0,1\}^\infty$ using this bijection.

Signature Schemes

- A tuple $\Pi = (Gen, Sign, Vrfy)$ of algorithms is a *signature scheme* if:
 - #1- *Gen*:
 - Is a probabilistic algorithm.
 - Takes as input a positive natural n .
 - Outputs a public/private key pair (pk, sk) denoted by the r.v. $Gen(n)$.
 - Is polynomial time in n .

Signature Schemes

- A tuple $\Pi = (Gen, Sign, Vrfy)$ of algorithms is a *signature scheme* if:
 - #2- *Sign*:
 - Is a probabilistic algorithm.
 - Seeks oracle access to an n -hash h .
 - Takes as input a private key sk and a message m .
 - Outputs a signature σ denoted by the r.v. $Sign_{sk}^h(m)$.
 - Is polynomial time in n .

Signature Schemes

- A tuple $\Pi = (Gen, Sign, Vrfy)$ of algorithms is a *signature scheme* if:
 - #3- *Vrfy*:
 - Is a deterministic algorithm.
 - Seeks oracle access to an n -hash h .
 - Takes as input a public key pk , a message m , and a signature σ .
 - Outputs a single bit 1 or 0 denoted by $Vrfy_{pk}^h(m, \sigma)$.
 - Is polynomial time in n .

Signature Schemes

- A tuple $\Pi = (Gen, Sign, Vrfy)$ of algorithms is a *signature scheme* if:
- #4- For every message m , for every variable-hash H , for every natural $n \in \mathbb{N}^+$, for every (pk, sk) in the support of $Gen(n)$ we have:

$$Vrfy_{pk}^{H_n}(m, Sign_{sk}^{H_n}(m)) = 1.$$

Honest Execution

- Step Zero: Everyone agrees to use a common security parameter n on a public variable-hash H .
- Step One: Signer announces the public key pk realized by $Gen(n)$ while keeping the secret key sk to themselves.
- Step Two: Signer chooses a desired message m and attaches the signature σ realized by $Sign_{sk}^{H_n}(m)$ to m before announcing it.
- Step Three: Verifiers trust the signer sent m iff running $Vrfy_{pk}^{H_n}(m, \sigma)$ yields 1.

Adversaries

- An algorithm \mathcal{A} is an *adversary* if it:
 - Is a probabilistic algorithm.
 - Seeks oracle access to a n -hash h as well as a signature function $\Sigma : \{0,1\}^* \rightarrow \{0,1\}^*$.
 - Takes as input a public key pk .
 - Outputs a message/signature pair (m, σ)
 - Is polynomial time in n .

Adaptive Chosen-Message Attack

- Suppose everyone has agreed to use a common security parameter n and public variable-hash H .
- Given an adversary \mathcal{A} and a signature scheme Π , define the probabilistic algorithm $SigForge_{\mathcal{A},\Pi}(n, H)$ as follows:
 - First store the pair (pk, sk) realized by $Gen(n)$.
 - Next \mathcal{A} obtains oracle access to H_n as well as $Sign_{sk}^{H_n}(_)$ and is given pk as input.
 - Let \mathcal{Q} be the set of messages \mathcal{A} queries to the oracle $Sign_{sk}^{H_n}(_)$ during its execution.
 - After the adversary terminates and outputs (m, σ) , check to see if $m \in \mathcal{Q}$. If so return 0.
 - Otherwise, check to see if $Vrfy_{pk}^{H_n}(m, \sigma) = 1$. If so return 1. Otherwise return 0.
- Roughly if $SigForge_{\mathcal{A},\Pi}(n, H) = 0$ then \mathcal{A} , even after receiving examples of valid signatures on many other messages, was not able to successfully forge a signature on a new message of its own choosing.

EUF-ACMA Security

- Say a signature scheme Π is “existentially unforgeable under an adaptive chosen-message attack” or, more simply, ***EUF-ACMA secure relative to a variable-hash H*** if for all adversaries \mathcal{A} and all $d \in \mathbb{N}^+$, for sufficiently large n we have:

$$P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) \leq \frac{1}{n^d}.$$

- EUF-ACMA security relative to a variable-hash H is difficult to prove but it entails many other kinds of security.

The Random Oracle Model

- Now, say \mathbb{H} is a r.v. which follows the uniform distribution over $\{0,1\}^\infty$.
- We can also view \mathbb{H} as a r.v. uniformly distributed over all variable-hashes.
- A signature scheme Π is ***EUF-ACMA secure in the random oracle model*** if for all adversaries \mathcal{A} and all $d \in \mathbb{N}^+$, for sufficiently large n we have:

$$P(\text{Sigforge}_{\mathcal{A},\Pi}(n, \mathbb{H}) = 1) \leq \frac{1}{n^d}.$$

- EUF-ACMA security in the random oracle model (ROM) is much easier to prove than EUF-ACMA security relative to a variable-hash H .
- Under standard assumptions, Bellare & Rogaway 1993 proved RSA-FDH is EUF-ACMA secure in the ROM.
- Still open whether a polynomial time variable-hash H exists relative to which RSA-FDH is EUF-ACMA secure...

Instantiating the ROM

- Question: If Π is proved EUF-ACMA secure in the ROM can we always find **some** variable-hash H relative to which Π is EUF-ACMA secure?
- Tadaki & Doi 2015: “**Yes!** Algorithmic randomness is the perfect tool.”
- Follow Up: What about a *computable* H ?
- Tadaki & Doi 2015: “We don’t know...”
- This Talk 2025: “**Yes!** Resource bounded randomness is the perfect tool.”

Instantiating the ROM

- Another Follow Up: What about a *polynomial time computable* H ?
- Canetti, Goldreich, & Halevi 2002: “No!”
- This Talk 2025: “I would bet that there is an elementary f such that we can find a $O(f(n))$ time computable H .”
- Final Question: What’s the *best* complexity guarantee we can establish?
- This Talk 2025: “I don’t know...perhaps *you have intuitions!*”

Algorithmic Randomness

- What makes an element of $\{0,1\}^\infty$ *random*?
- Given a r.v. like \mathbb{H} , probability theory helps us gauge the relative likelihood its outcomes.
- However, probability theory does not capture all our intuitions about randomness.
- Consider the following sequences:

10101010101010...

01001101110100...

- When pressed most would say the second one is ‘more random’ than the first. Why?
- The first has a rare property of never seeing two 0’s in a row (a probability 0 event) while the second one does not.
- So maybe a sequence is random iff one cannot specify a rare property it possesses.
- But then all sequences have the rare property of being themselves...so no sequence is random?

Algorithmic Randomness

- Algorithmic randomness responds as follows:

An element of $\{0,1\}^\infty$ is random iff one cannot effectively specify a sequence of rarer and rarer properties that it possesses.

- Many ways of cashing out this answer and traditional approaches take ‘effectively’ to mean ‘computably.’
- As a result, traditional algorithmic randomness notions require that random elements of $\{0,1\}^\infty$ be uncomputable.

Tadaki & Doi's Meta-Question

- Cryptography asks: which variable-hashes H can instantiate the security provided by \mathbb{H} ?
- Algorithmic randomness asks: which elements of $\{0,1\}^\infty$ are random realizations of \mathbb{H} ?
- Tadaki & Doi 2015 ask: are these the same question?

Tadaki & Doi's Result

- Martin-Löf (ML) randomness is often taken to be the 'one true' algorithmic randomness notion.
- Tadaki & Doi 2015 show if a variable-hash H is Martin-Löf random then *every* scheme Π which is proved EUF-ACMA secure in the ROM is also EUF-ACMA secure relative to H .
- Note, H is *scheme agnostic*.
- Sadly however, Martin-Löf randoms are uncomputable.
- They note Schnorr randoms also work, but these too are uncomputable.

Tadaki & Doi's Conjecture

- Tadaki & Doi 2015 conjectured that every scheme Π which is proved EUF-ACMA secure in the ROM is also EUF-ACMA secure relative to some *computable* variable-hash H .
- I have recently shown that this conjecture is true using resource bounded randomness!

Resource Bounded Randomness

- Recall our pretheoretic definition of randomness:

An element of $\{0,1\}^\infty$ is random iff one cannot effectively specify a sequence of rarer and rarer properties that it possesses.

- Resource bounded randomness researchers say there is no real reason to cash out ‘effectively’ as ‘computably.’
- ‘Effectively’ could mean ‘primitive recursively’ or ‘exponential time computably’ or ‘polynomial space computably’ etc.
- This means, **resource bounded randomness notions can have computable instances.**
- I believe resource bounded randomness is the correct paradigm for studying the scheme agnostic instantiation of cryptographic security.

Primitive Recursive Schnorr Randomness

- A sequence of subsets $\{U_i\}_{i \in \mathbb{N}^+}$ of $\{0,1\}^\infty$ is *primitive recursively approximable* if there is an array of subsets $\{S_{i,j}\}_{(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+}$ of $\{0,1\}^*$ such that:
 - #1- There is a primitive recursive function $f : \mathbb{N}^+ \times \mathbb{N}^+ \rightarrow \mathbb{N}^+$ so $\forall (i,j) \in \mathbb{N}^+ \times \mathbb{N}^+$ we have:

$$\max\{ |w| : w \in S_{i,j} \} \leq f(i,j).$$

Primitive Recursive Schnorr Randomness

- A sequence of subsets $\{U_i\}_{i \in \mathbb{N}^+}$ of $\{0,1\}^\infty$ is *primitive recursively approximable* if there is an array of subsets $\{S_{i,j}\}_{(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+}$ of $\{0,1\}^*$ such that:
 - #2- The language $L = \{\langle w, i, j \rangle : w \in S_{i,j}\}$ is primitive recursive, i.e. there is a primitive recursive algorithm which decides membership in L .

Primitive Recursive Schnorr Randomness

- A sequence of subsets $\{U_i\}_{i \in \mathbb{N}^+}$ of $\{0,1\}^\infty$ is *primitive recursively approximable* if there is an array of subsets $\{S_{i,j}\}_{(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+}$ of $\{0,1\}^*$ such that:
 - #3- $\forall i \in \mathbb{N}^+$ we have:

$$U_i = \bigcup_{j \in \mathbb{N}^+} [S_{i,j}]$$

where $[S_{i,j}]$ denotes all the infinite extensions of strings in $S_{i,j}$.

Primitive Recursive Schnorr Randomness

- A sequence of subsets $\{U_i\}_{i \in \mathbb{N}^+}$ of $\{0,1\}^\infty$ is *primitive recursively approximable* if there is an array of subsets $\{S_{i,j}\}_{(i,j) \in \mathbb{N}^+ \times \mathbb{N}^+}$ of $\{0,1\}^*$ such that:
 - #4- $\forall (i,j) \in \mathbb{N}^+ \times \mathbb{N}^+$ we have:

$$\lambda \left(U_i - \bigcup_{k=1}^j [S_{i,k}] \right) \leq 2^{-j}.$$

Primitive Recursive Schnorr Randomness

- A *primitive recursive Schnorr test* is a descending sequence of subsets $\{U_i\}_{i \in \mathbb{N}^+}$ of $\{0,1\}^\infty$ such that $\{U_i\}_{i \in \mathbb{N}^+}$ is primitive recursively approximable and $\lambda(U_i) \leq 2^{-i}$.
- A sequence $\alpha \in \{0,1\}^\infty$ passes a primitive recursive Schnorr test $\{U_i\}_{i \in \mathbb{N}^+}$ if $\alpha \notin \bigcap_{i \in \mathbb{N}^+} U_i$.
- A sequence $\alpha \in \{0,1\}^\infty$ is *primitive recursive Schnorr random* if it passes all primitive recursive Schnorr tests.

Proof Strategy

- Henceforth, suppose Π is EUF-ACMA secure in the ROM:
 - Π is *not* EUF-ACMA secure relative to H iff there exists an adversary \mathcal{A} and $d \in \mathbb{N}^+$ so that for infinitely many n :

$$P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) > \frac{1}{n^d}.$$

- We will show for all adversaries \mathcal{A} and $d \in \mathbb{N}^{>1}$ there exists $N \in \mathbb{N}^+$ so that the sequence of sets:

$$U_i = \bigcup_{n \geq N+2^{i+1}} \left\{ H : P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) > \frac{1}{n^d} \right\}$$

is a primitive recursive Schnorr test.

- It follows that if H is primitive recursive Schnorr random then Π is EUF-ACMA secure relative to H .

Proof Strategy

- Lemma (Tadaki & Doi 2015)— Fix an adversary \mathcal{A} and $d \in \mathbb{N}^{>1}$. There exists an $N \in \mathbb{N}^+$ so that $\forall M \geq N$:

$$\lambda \left(\bigcup_{n > M} \left\{ H : P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) > \frac{1}{n^d} \right\} \right) \leq \frac{2}{M}.$$

- Thus, if we define:

$$U_i = \bigcup_{n \geq N + 2^{i+1}} \left\{ H : P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) > \frac{1}{n^d} \right\}$$

then:

$$\lambda(U_i) \leq \frac{2}{2^{i+1} + N} \leq 2^{-i}.$$

- Since $\{U_i\}_{i \in \mathbb{N}^+}$ is clearly descending, all that remains is to show $\{U_i\}_{i \in \mathbb{N}^+}$ is primitive recursively approximable.

Constructing an Array

- Since $Sign$, $Vrfy$, and \mathcal{A} are all polynomial time in n , we can upper bound the maximum of their running times by some polynomial $q(n)$.
- This also means for all variable-hashes H , we can upper bound the length of inputs given to H_n while running $SigForge_{\mathcal{A},\Pi}(n, H)$ by $q(n)$.
- Thus to determine whether:

$$P(Sigforge_{\mathcal{A},\Pi}(n, H) = 1) > \frac{1}{n^d}$$

it suffices for H_n to be defined only on inputs of length less than or equal to $q(n)$.

Constructing an Array

- Define the primitive recursive algorithm $reqLen(n)$ to return the minimum length required for a string w so that given any two infinite extensions w^1 and w^2 , the n -hashes w_n^1 and w_n^2 are identical on all inputs of length less than or equal to $q(n)$.
- Next, define the primitive recursive algorithm $decideL(w, i, j)$ as follows:
 - For n in $[N + 2^{i+j}, N + 2^{i+j+1})$:
 - If $|w| = reqLen(n)$ and $P(Sigforge_{\mathcal{A}, \Pi}(n, w\bar{0}) = 1) > \frac{1}{n^d}$, return 1.
 - Otherwise, return 0.
- Finally, define $S_{i,j}$ so that $w \in S_{i,j}$ iff $decideL(w, i, j) = 1$.

Constructing an Array

- Note:

- #1- $\max\{|w| : w \in S_{i,j}\} \leq \max\{reqLen(n) : n \in [N + 2^{i+j}, N + 2^{i+j+1})\}$.

- #2- $decideL(w, i, j)$ decides whether $w \in S_{i,j}$.

- #3- $U_i = \bigcup_{j \in \mathbb{N}^+} [S_{i,j}]$ as:

$$[S_{i,j}] = \bigcup_{n \in [N+2^{i+j}, N+2^{i+j+1})} \left\{ H : P(\text{Sigforge}_{\mathcal{A}, \Pi}(n, H) = 1) > \frac{1}{n^d} \right\}.$$

- #4- $\lambda \left(U_i - \bigcup_{k=1}^j [S_{i,k}] \right) \leq \lambda(U_{i+j}) \leq 2^{-(i+j)} \leq 2^{-j}$.

- In other words, $\{U_i\}_{i \in \mathbb{N}^+}$ is primitive recursively approximable!

Taking Stock

- Recap: we just showed that if a variable-hash H is primitive recursive Schnorr random then *every* scheme Π which is proved EUF-ACMA secure in the ROM is also EUF-ACMA secure relative to H .
- Why do we know there exist computable primitive recursive Schnorr randoms?
- To answer this we now present a *martingale characterization* of primitive recursive Schnorr randomness.

Martingale Characterization

- A function $m : \{0,1\}^* \rightarrow \mathbb{R}^{\geq 0}$ satisfies *the martingale property* if $m(\varepsilon) > 0$ and for all $w \in \{0,1\}^*$:

$$m(w) = \frac{m(w0) + m(w1)}{2}.$$

- Let $\mathbb{Q}_2^{\geq 0}$ denote the set of non-negative dyadic rationals.
- A *primitive recursive martingale* is a primitive recursive function $m : \{0,1\}^* \rightarrow \mathbb{Q}_2^{\geq 0}$ that satisfies the martingale property.
- An *order* $h : \mathbb{N} \rightarrow \mathbb{N}$ is an unbounded non-decreasing function.
- Given an order h , we define its *inverse* $\text{inv}_h(n)$ to return the smallest k for which $h(k) \geq n$.
- A *true primitive recursive order* is an order such that both itself and its inverse are primitive recursive.

Martingale Characterization

- Theorem (Sureson 2017): A sequence $\alpha \in \{0,1\}^\infty$ is primitive recursive Schnorr random iff for every primitive recursive martingale m and every true primitive recursive order h we have that $m(\alpha \upharpoonright n) \geq 2^{h(n)}$ for only finitely many n .

Putnam's Ghost

- Fact: One can recursively enumerate all primitive recursive martingales.
- Let m_i denote the i -th primitive recursive martingale in this enumeration.
- Let $m^* = \sum_{i \in \mathbb{N}^+} 2^{-i} \cdot \frac{m_i}{m_i(\varepsilon)}$.
- m^* is a computable function which satisfies the martingale property.
- We recursively construct a computable sequence β on which m^* is bounded as follows:

$$\beta(n) = \begin{cases} 1 & m^*(\beta(0) \dots \beta(n-1)0) \geq m^*(\beta(0) \dots \beta(n-1)) \\ 0 & m^*(\beta(0) \dots \beta(n-1)1) > m^*(\beta(0) \dots \beta(n-1)) \end{cases}$$

- Since m^* is bounded on β , each m_i is also bounded on β .
- Therefore, β is a computable primitive recursive Schnorr random.

Conclusion

- Resource bounded randomness allows us to study the scheme agnostic instantiation of cryptographic security.
- Useful for the randomness notion to admit a test characterization as well as a martingale characterization.
- Test characterizations allow us to easily argue that the randomness notion suffices for instantiating cryptographic security.
- Martingale characterizations allow us to easily argue that the randomness notion has efficiently calculable instances.
- Future work should see if our results can be extended beyond primitive recursive Schnorr randomness...perhaps **polynomial space randomness**?

End

This would let us find an $O(f(n))$ time computable H !

(Where f is elementary)