T-79.159 Cryptography and Data Security

Lecture 9: Pseudorandomness, Provable Security

Helger Lipmaa
Helsinki University of Technology
helger@tcs.hut.fi

Security Notions. Provable Security

- Definitional approach: First *define* what do you mean by security

- . . . Correct definition is vital

- Thereafter *construct* a primitive that satisfies the definition

- Construction of primitive B is often based on some other primitive A that satisfies some other definition

  ⋆ Familiar reduction arguments: If A (is secure) and $A \Rightarrow B$ then B (is secure). If $\neg B$ and $A \Rightarrow B$ then $\neg A$

Security Notions. Provable Security

- Construction of primitive B is often based on some other primitive A that satisfies some other definition

  ⋆ Familiar reduction arguments: If A (is secure) and $A \Rightarrow B$ then B (is secure). If $\neg B$ and $A \Rightarrow B$ then $\neg A$

- Recall NP-completeness: If A is NP-complete and from an “efficient” algorithm b, solving B, one can deduce an polynomial-time algorithm a (that uses b as a subroutine) that solves A, then also B is NP-complete

- Same logic in provable security, but *reductions must be tight*

Ideal block cipher = Random permutation

- What is the most secure block cipher in this world?
- Answer: a family of random permutations

Random permutation (RP)

- Fix $\mathcal{P}$, $\mathcal{K}$, $\mathcal{C}$. Let Perm be the set of all permutations $f : \mathcal{P} \longrightarrow \mathcal{C}$

- Random permutation: a randomly chosen permutation from Perm

- Permutation: if you have seen $f(x)$, seeing $f(x)$ again does not give any new information

- Random: if you have not seen $f(x)$, you have no better strategy than to guess the value $f(x)$, except that it must not be equal to $f(y)$ for $f(y)$ that you have seen before

Random function (RF)

- Used when the cipher does not have to be bijective (e.g., stream ciphers)
- Random function = randomly chosen function
- If you have seen $f(x)$, you already know it
- If you have not seen $f(x)$, your best strategy is to guess $f(x)$ randomly

Family of random permutations

- Let $k \in \mathcal{K}$ index a random permutation $f \in \mathsf{Perm}$
- Block cipher is a family of permutations, indexed by keys
- Random (block) cipher is a family of random permutations
- I.e., E_{k_1} and E_{k_2} are independent and random permutations when $k_1 \neq k_2$
- Example: OTP has $\mathcal{K} = \{0, 1\}$, E_0 is a permutation $(01) \to (01)$, E_1 is a permutation $(10) \to (01)$

Ideal ciphers: hazards

- Implementing requires a database of $|\mathcal{P}| \geq 2^{64}$ values
- The key corresponds one-to-one to the permutation, so $|\mathcal{K}| = |\mathcal{P}|!$, and one needs $\log_2 |\mathcal{P}|! \approx |\mathcal{P}| \log_2 |\mathcal{P}|$ bits to transport $|\mathcal{K}|$
- Less efficient than the OTP! (Why?)
- So we need something more practical...

Computational security

- Unconditional security: function *is* random, bitstring *is* random
- Computational security: function *seems to be* random, bitstring *seems to be* random
- ... to an adversary who has limited resources
- Limited = polynomial-time (in *security parameter* k, usually the key length) or in general, works in time $t(k)$ for some function t

Pseudorandom permutations: Preliminaries

- PRP: a permutation that *looks like* a RP to a poly-time bounded adversary
- Let f be a family of permutations, $f : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$
- Let X be a random variable (it might be output of an randomized algorithm) with a known distribution
- $x \leftarrow_R X$ denotes that x is chosen to be the value of the random variable X, according to this distribution
- $k \leftarrow_R \mathcal{K}$ — k is a random element from the set $\mathcal{K}$ (often uniform)

Oracle model (1/2)

- Oracle = subroutine, accessed in a black-box mode
- . . . I.e., can give some inputs and receive corresponding outputs
- . . . No access to the internals to oracle!

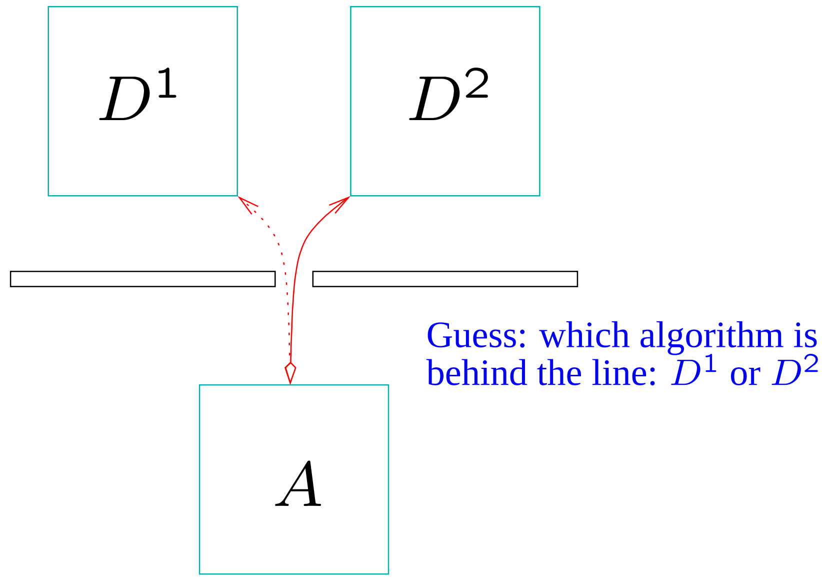
Oracle model (2/2)

- Oracle can be plugged in to another algorithm, exactly like a subroutine can be referenced by a pointer

- Denoted: A^B (A uses B as an oracle)

- A calls the subroutine/queries the oracle q times

Distinguishing



- A *ε-distinguishes* D^1 and D^2 if $|\Pr[x \leftarrow_R D^1 : A(x) = 2] - \Pr[x \leftarrow_R D^2 : A(x) = 2]| \geq \varepsilon$.

Definition of an PRP

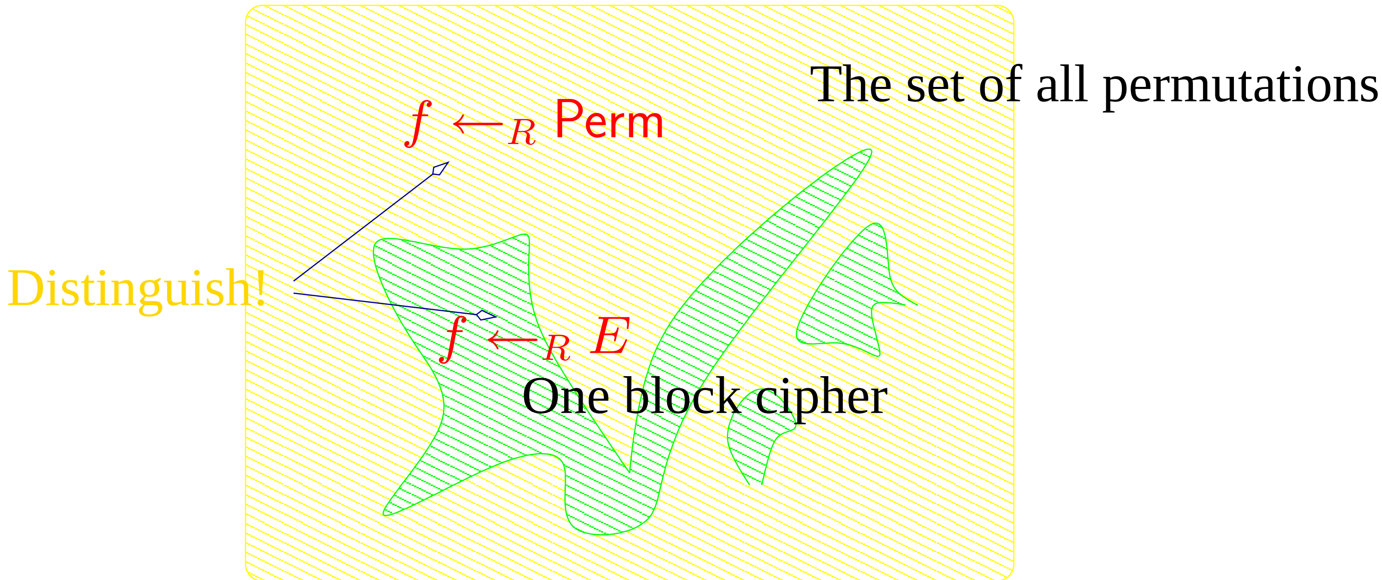
Fix k, the key length. Let E be a family of permutations (i.e., a block cipher), and let Perm be the family of all permutations

Intuitively: A has a success probability ε against a block cipher E, if it can distinguish E_K, with a random key, from the random permutation.

Let A be an algorithm. Define its success probability against the PRP E to be

$$\begin{aligned}\mathsf{Succ}_E^{\mathsf{PRP}}(A) := |\Pr_k[f \leftarrow_R E : A^f(k) = 1] \\ - \Pr_f[f \leftarrow_R \mathsf{Perm} : A^f(k) = 1]| \enspace .\end{aligned}$$

Picture: PRP definition



(In reality, the green area should be really really small)

Definition of an PRP

- We say that E is an (q, t, ε)-*secure PRP* if for any algorithm that spends at most t steps (in some well-defined machine model), queries the oracle at most q times, has the success probability $\leq \varepsilon$ of distinguishing E, $\mathsf{Succ}_f^{\mathsf{PRP}}(A) \leq \varepsilon$.

- The same adversary can achieve larger success probability if q and t are increased. Thus $\varepsilon = \varepsilon(q, t)$.

Formal Def: Symmetric Cryptosystems

- Symmetric cryptosystem Π = *pseudo-random function* from $\{0,1\}^n \to \{0,1\}^{p(n)}$ for some polynomial p

- Security definition: consider a distinguishing game as in the case of PRPs, but now we have a randomly chosen permutation is replaced with a randomly chosen function

- Symmetric cryptosystem Π is (q, μ, t, ε)-*secure*, if it cannot be ε-distinguished by any algorithm that works in time t and makes no more than q queries, with in total μ blocks of queried plaintext

Symmetric Cryptosystems: Constructions

- Standard construction: A block cipher (a (q, t, ε)-secure PRP) + a good block cipher mode

- Block ciphers: security is heuristic

- But reduction must still be tight

Block cipher modes: Security

- When proving security, assume that first you have an ideal block cipher (RP) with the concrete mode. Prove that then the cryptosystem is $(q_1, \mu_1, t_1, \epsilon_1)$ secure
- This gives you an idea of how much security can be achieved at all with this mode
- Substitute RO with a (q_2, t_2, ϵ_2)-secure PRP. Prove that the resulting cryptosystem is $(q_3, \mu_3, t_3, \epsilon_3)$-secure for ϵ_3
- Give *tight* proofs: exhibit an adversary that meets the bound

Security of CBC mode

Theorem Let $E : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$ be an $(q_1, t_1, \varepsilon_1)$-secure PRP. The cryptosystem $\mathsf{CBC} - E$ (E used in conjunction with CBC mode) is then $(q_2, \mu, t_2, \varepsilon_2)$ secure for some (q_2, t_2), where $\mu = q_1 \ell$ and $\varepsilon_2 = \varepsilon_1 + \frac{2\mu^2}{\ell^2 2^\ell}$.

This means that when using a secure block cipher with the CBC mode, then one can must have $\mu^2 \ll 2^\ell$ for the cryptosystem to be secure.

In other words: If the block length is ℓ bits then you can encrypt up to $2^{\ell/2}$ block with the CBC mode and still feel secure. The same holds for the CTR mode. Reason: *birthday paradox*

The term $2^{\ell/2}$ in security of CTR

- Idea: can't reuse the keystream (affects security)

- What is the probability of reusing the keystream if ctr is chosen randomly?

- If ctr is maintained as a state and always increased, the keystream is never reused. Can encrypt 2^ℓ blocks!

- If ctr is chosen randomly, one has birthday paradox:

- ...after $\sqrt{2^\ell} = 2^{\ell/2}$ blocks, some part of the keystream is reused with a high probability

Importance of exact reductions

- We gave an exact reduction for the security of the CBC mode
- Thanks to that we know that encrypting more than $2^{\ell/2}$ bits by using the same key might be harmful
- Now, ℓ is a fixed parameter: say, $\ell = 64$ (in the case of DES)
- In the case of "usual" complexity-theoretic reductions, you would know that encrypting more than $p(\ell)$, p some polynomial, bits is harmful

Importance of exact reductions

- Bad, since:
  - ⋆ You usually do not know p — and (say) ℓ^4 and ℓ^2 give *very* different safety bounds (2^{24} and 2^{12} bits, respectively),
  - ⋆ Polynomial $p(\ell)$ is often a very small number
    - ∗ If we would know that we can securely encrypt ℓ^4 bits, this would make 2^{28} if $\ell = 128$, while with safety bound $2^{\ell/2}$, we can encrypt 2^{64} bits!
- Holy Grail of provable security: Give tight reductions for existing constructions, find new (efficient) constructions with even tighter restrictions

How to construct PRPs, PRFs?

- We know how to build cryptosystems, based on secure PRPs
- How to construct PRPs themselves?
- Is it an abstaction like a RP or can it be constructed?
- It *can* be constructed, but this requires tools from complexity theory and number theory

Naor-Reingold Number-Theoretic PRF Generator

- Group-theoretic setting (again): Primes q, p, $q \mid (p-1)$. Let g be an element of $\mathbb{Z}_p^*$, with order q, let G be the subgroup generated by g

- Let $\vec{a} = (a_0, \dots, a_n) \in \mathbb{Z}_q^{n+1}$

- For any key $K = (p, q, g, \vec{a})$, and any input $x = x_1 \dots x_n$, define
$$f_K(x) := (g^{a_0})^{\prod_{x_i=1} a_i} \ .$$

- Define F_n to be the distribution induced when one chooses (some) n-bit prime p, (some) large prime divisor q of $p - 1$ and (some) element g of order q in $\mathbb{Z}_p^*$, and a (*random*) element $\vec{a}$ of $\mathbb{Z}_q^{n+1}$.

Naor-Reingold Number-Theoretic PRF Generator

- Naor, Reingold: the described construction is a secure PRF generator if the Decisional Diffie-Hellman assumption holds

- That is, a polynomial-time adversary cannot distinguish a random member of F_n from a random function $\{0,1\}^n \to G$

Reminder: Distributions

- Uniform probability distribution U_n on $\{0, 1\}^n$: if X follows U_n then

$$\Pr[X = x] = 2^{-n} \quad \text{if } |x| = n.$$

- Support of a distribution D = set of elements x that have nonzero probability

- Let D, E be families of distributions, such that the support of D_n, E_n is a subset of $\{0, 1\}^n$

- $x \leftarrow_R D_n$ — x is drawn from $\{0, 1\}^n$ according to D_n

Pseudorandom generator

- Let $f : \{0,1\}^n \to \{0,1\}^m$, $m > n$, be an efficient algorithm

- Define $\mathsf{Succ}_f^{\mathsf{PRG}}(A) := |\Pr[x \leftarrow_R U_m : A(x) = 1] - \Pr[x \leftarrow_R f(U_n) : A(x) = 1]|$

- I.e.: A is successful if she distinguishes the output of f (keystream) on an uniformly distributed short input (seed) from a uniformly distributed long string

- f is a (t, ε)-*secure pseudorandom generator* if no A that takes $\leq t$ steps has $\mathsf{Succ}_f^{\mathsf{PRG}}(A) \geq \varepsilon$

Synchronous stream cipher = PRG

- Objective of a s. stream cipher: The output of G (keystream) on an uniformly distributed short input (seed) should be indistinguishable from a uniformly distributed long string

- Thus, a synchronous stream cipher can be modeled as a (t, ε)-secure pseudorandom generator (PRG) G, with $E_K(x) = x \oplus G(K)$, where $|K| = n$ and $|x| = m$

- Ideally: t "big" ($\approx 2^n$), ε small ($\approx 2^{-n}$)

- If we omit (t, ε) we usually assume that t is very big and ε is very small

Block and stream ciphers

- Block cipher: family of permutations, $E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$

Ideally Modeled by families of pseudorandom permutations

- (Synchronous) stream cipher: key stream function G

Ideally Modeled by *pseudorandom generators*

Reminder: One-way functions

- Intuition: it is easy to compute f, but hard to invert it
- Example: (1) multiplication of two numbers. Easy to multiply, hard to factor; (2) exponentiation in a subgroup G of order q in $\mathbb{Z}_p^*$, where $q \mid (p-1)$ and q, p are primes. Easy to compute g^x, hard to find x (*discrete logarithm*), given (g, g^x)
- Thus, there seem to be natural candidates for OWFs
- Formally: $\mathsf{SuccOWF}_f(A) = \Pr[f(A(f(x)) = x]$
- One-way permutation: Permutation that is an OWF

OWF $\Rightarrow$ PRG

For $x, r \in \{0,1\}^n$, define $x \cdot r = x_1 r_1 + \cdots + x_n r_n$ to be their dot product

Theorem (Impagliazzo, Levin, Luby, 1989) Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation. Let $x, r \leftarrow_R U_n$. Then $g : \{0,1\}^n \to \{0,1\}^{2n+1}$,

$$g(x) = f(x) || r || x \cdot r$$

is a (t, ε)-pseudorandom generator for reasonable (t, ε)

One can also construct a PRG given any OWF (the same paper)

Thus, we can construct a PRG, given the existence of an OWF

OWF $\Rightarrow$ PRF $\Rightarrow$ PRP

- Goldreich, Goldwasser, Micali (1984): A PRF can be constructed from any PRG
- Luby, Rackoff (1988): A PRP can be constructed from any PRF (Feistel ciphers)
- Opposite direction also holds! (block cipher modes)
- Combining these results: block ciphers and stream ciphers exist exactly if one-way functions exist. There are efficient algorithms for transforming a secure stream cipher to a secure block cipher, and vice versa

Caveats

- Efficiency: known candidates of OWF are severely less efficient than AES and other efficient block and stream ciphers

- Provable security comes at the expense of efficiency!

  ⋆ At least currently: it is not known how to prove the security of of efficient block and stream ciphers

- Security: It is *not* known if one-way functions exist, although it is strongly conjectured that this is the case