CS4501 ~~Cryptographic Protocols~~
Intro to Cryptography Speedrun
Lecture 13: Discrete Logarithms, Diffie-Hellman Key Exchange

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Roadmap of the Next Few Lectures

- Introducing Oblivious Transfer.
- Bounded Adversaries and Computational Security.
- The Cryptographic Toolkit: One-Way Functions.
- The Cryptographic Toolkit: Assumptions on Groups.
- A Simple Application: Key Agreement.
- Ensembles and Computational Indistinguishability.
- Computational Security for Protocols.
- The Cryptographic Toolkit: Public-key Encryption.
- Realizing Oblivious Transfer.

Recap: Asymptotic Security

- Before we start thinking about distinguishing distributions, we're going to talk about the simpler case of playing *security games* with the adversary.
- We will define a *security parameter* $\kappa \in \mathbb{N}$ and assume that the adversary's computing power to grow polynomially in κ. We want the probability that it wins a security game to diminish *faster than that.*
- If we assume that honest parties adjust κ to match their increasing compute budget over time, this leads to a notion of cryptography that gets *more secure* over time, even though the adversary's power increases faster than the honest parties' power.

Definition 1 (PPT): An algorithm A runs in *probabilistic polynomial time* (PPT) if there exists a polynomial p such that for every input $x \in \{0,1\}^*$ and every random tape $r \in \{0,1\}^*$, the computation of $A(x; r)$ halts within $p(|x|)$ steps.

Definition 2 (negligible fn): A function ε is *negligible* if $\forall c \in \mathbb{N}\ \exists x_0 \text{ s.t. } \varepsilon(x) < x^{-c}$. In other words, a negligible function diminishes faster than *any* inverse polynomial.

Definition 3: The One-Way Function Game

- The *one-way function game* $\mathsf{OWFGame}_{f,\mathcal{A}}(\kappa)$ is played between a "challenger" and the adversary. The output of the game is the output of the challenger. The game is as follows:

$x \leftarrow \{0,1\}^\kappa$

$y := f(x)$

$1^\kappa, y$

$x' \in \{0,1\}^\kappa$

outputs 1 if $f(x') = y$
outputs 0 otherwise

One-Way Functions

Definition 3 (OWF): $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* (OWF) if:

1. f is PPT.
2. $\forall$ PPT $\mathcal{A}$ $\exists$ a negligible function ε such that $\forall \kappa \in \mathbb{N}$, $\Pr[\mathsf{OWFGame}_{f,\mathcal{A}}(\kappa) = 1] \leq \varepsilon(\kappa)$

- Intuitively, f is easy to compute, but very hard to invert.
- Importantly, this holds for *average-case* (i.e. random) inputs.

$\mathsf{OWFGame}_{f,\mathcal{A}}(\kappa)$

$x \leftarrow \{0,1\}^\kappa$

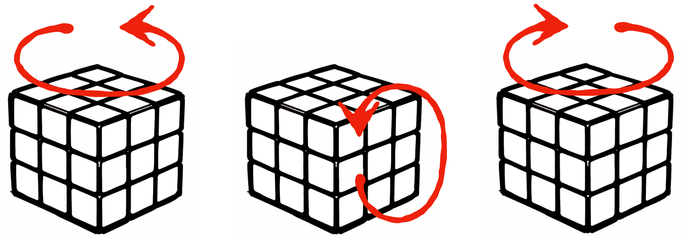
$y := f(x)$

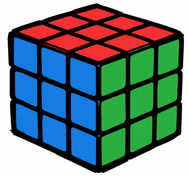
$1^\kappa, y$ →

← $x' \in \{0,1\}^\kappa$

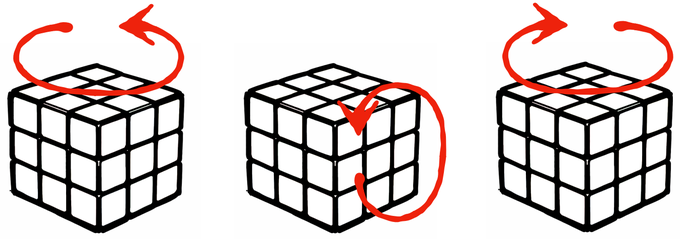
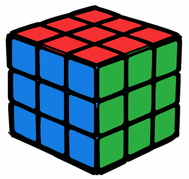
outputs 1 if $f(x') = y$
outputs 0 otherwise

Beware of Analogies

It's very easy to sample a random $x =$ 

and apply it to  in order to get $y =$  .

It's much harder to start from $y =$  and work out an

$x' =$  that you can do in reverse to get  .

Recap: More Group Theory

Definition 4 (group exponentiation): Let $(\mathbb{G}, \cdot)$ be a group, let $g \in \mathbb{G}$ be any element, and let $g^{-1} \in \mathbb{G}$ be its inverse. Let $1_{\mathbb{G}}$ be the identity element of $(\mathbb{G}, \cdot)$. We will define $g^0 = 1_{\mathbb{G}}$, and for $x \in \mathbb{N}$ we will define $g^x = g \cdot g^{x-1}$ and $g^{-x} = g^{-1} \cdot g^{-(x-1)}$.

Proposition 1: Let $(\mathbb{G}, \cdot)$ be a group, $g \in \mathbb{G}$, and $x, y \in \mathbb{Z}$. We have $g^x \cdot g^y = g^{x+y}$ and $(g^x)^y = g^{x \cdot y}$.

Definition 5 (generated subgroups, generators): Let $(\mathbb{G}, \cdot)$ be a group and let $g \in \mathbb{G}$ be any element. The subgroup *generated* by g is $\langle g \rangle = \{g^x : x \in \mathbb{Z}\}$. We say g is the *generator* of $\langle g \rangle$.

Definition 6 (orders of group elements): Let $(\mathbb{G}, \cdot)$ be a group and let $g \in \mathbb{G}$ be any element. The *order* of g, denoted $\text{ord}(g)$, is the order of the subgroup generated by g; I.e., $\text{ord}(g) = |\langle g \rangle|$. $\text{ord}(g)$ is also the smallest $x \in \mathbb{N}$ such that $x > 0 \wedge g^x = 1_{\mathbb{G}}$. If no such x exists, then $\text{ord}(g) = \infty$.

Definition 7 (cyclic groups): A group $(\mathbb{G}, \cdot)$ is *cyclic* if it is generated by a single element. That is, if there exists some element $g \in \mathbb{G}$ such that $\langle g \rangle = \mathbb{G}$. Note that $\langle g \rangle$ is cyclic by definition.

Proposition 2: Let $(\mathbb{G}, \cdot)$ be a finite group such that $|\mathbb{G}| = m$. For every $g \in \mathbb{G}$, $g^m = g^0 = 1_{\mathbb{G}}$.

Proposition 3: If $(\mathbb{G}, \cdot)$ is a prime-order group, then it's cyclic, and every $g \in \mathbb{G} \backslash \{1_{\mathbb{G}}\}$ generates $\mathbb{G}$.

Proposition 4: If $(\mathbb{G}, \cdot)$ is a cyclic group, then it's also abelian (i.e. commutative).

Recap: More Group Theory

Definition 8 (group homomorphism): Let $(\mathbb{G}, \star)$ and $(\mathbb{H}, \diamond)$ be groups. A *homomorphism* from $\mathbb{G}$ to $\mathbb{H}$ is a function $f : \mathbb{G} \to \mathbb{H}$ such that for every $g_1, g_2 \in \mathbb{G}$, we have $f(g_1 \star g_2) = f(g_1) \diamond f(g_2)$. In other words, a homomorphism is a map from $\mathbb{G}$ to $\mathbb{H}$ that preserves the structure of $\mathbb{G}$.

Definition 9 (group isomorphism): Let $(\mathbb{G}, \star)$ and $(\mathbb{H}, \diamond)$ be groups. An *isomorphism* from $\mathbb{G}$ to $\mathbb{H}$ is a homomorphism that is both injective (one-to-one) and surjective (onto). If there exists an isomorphism between $\mathbb{G}$ and $\mathbb{H}$, then we call them *isomorphic*, and denote this $\mathbb{G} \cong \mathbb{H}$.

Intuitively, groups that are isomorphic have *the same structure*, even if their elements and operations are different, and structure-preserving maps exist in both directions.

Example: $(\mathbb{Z}_3^*, \cdot_3)$ and $(\mathbb{Z}_2, +_2)$ are isomorphic. Note that $\mathbb{Z}_3^* = \{1,2\}$ and $\mathbb{Z}_2 = \{0,1\}$.
Consider $f : \mathbb{Z}_3^* \to \mathbb{Z}_2$ such that $f(1) = 0$ and $f(2) = 1$. Clearly f is one-to-one and onto.
Let's verify by exhaustive inspection that it's a homomorphism:

$$f(1 \cdot_3 1) = f(1) = 0 = 0 +_2 0 = f(1) +_2 f(1)$$

$$f(1 \cdot_3 2) = f(2) = 1 = 0 +_2 1 = f(1) +_2 f(2)$$

$$f(2 \cdot_3 2) = f(1) = 0 = 1 +_2 1 = f(2) +_2 f(2)$$

Group Exponentiation is an Isomorphism

Lemma 1: Let $(\mathbb{G}, \cdot)$ be a cyclic group of order m, and let g be a generator of $\mathbb{G}$ (i.e. $\langle g \rangle = \mathbb{G}$ and $|\mathbb{G}| = m$). The mapping $f : \mathbb{Z}_m \to \mathbb{G}$ defined by $f(x) = g^x$ is an isomorphism.

Proof: We must show that f is a homomorphism that is one-to-one and onto.

- Homorphism: Let $x, y \in \mathbb{Z}_m$. We have $f(x + y) = g^{x+y} = g^x \cdot g^y = f(x) \cdot f(y)$, by Proposition 1.
- Injectiveness: Let $x, y \in \mathbb{Z}_m$. If $f(x) = f(y)$, then $g^x = g^y$.

  If g^{-y} is the inverse of g^y, then Proposition 1 ($g^x \cdot g^y = g^{x+y}$) yields $g^{x-y} = 1_{\mathbb{G}} = g^0$.

  If $z := (x - y) \bmod m$, then Propositions 1 and 2 ($g^m = g^0$) imply $g^{x-y} = g^z$.

  Since we know $0 \leq z < m$, and m is the *smallest* integer such that $m > 0 \wedge g^m = g^0$ (Definitions 5 - 7), it must be the case that $0 = z$.

  This implies $0 = (x - y) \bmod m$, then $x \bmod m = y \bmod m$, and finally $x = y$.
- Surjectiveness: Follows from the cyclicality of $(\mathbb{G}, \cdot)$. Given $h \in \mathbb{G}$, $\exists x \in \mathbb{Z}_m$ such that $h = g^x$. ∎

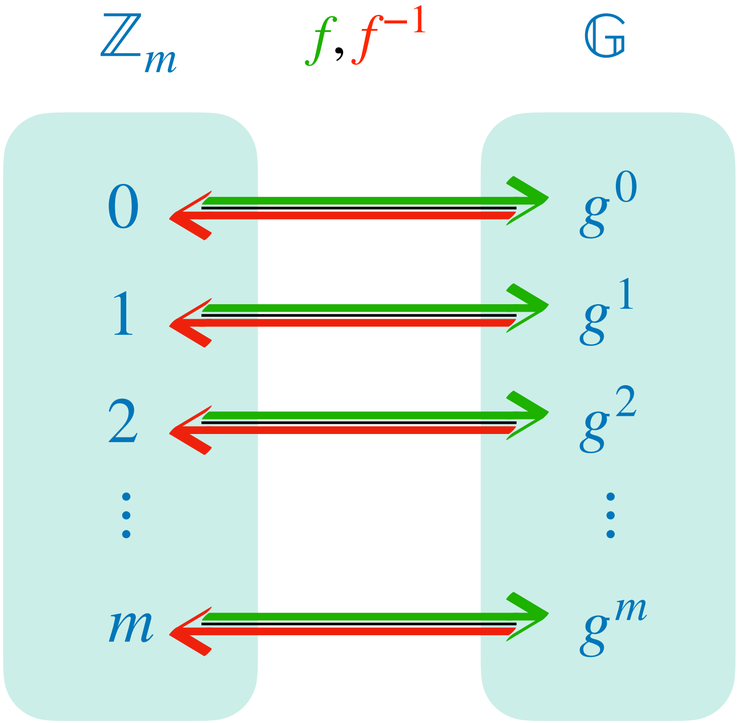
Corollary 1: Let $(\mathbb{G}, \cdot)$ be a cyclic group of order m, and let g be a generator of $\mathbb{G}$. If x is uniformly distributed in $\mathbb{Z}_m$, then g^x is uniformly distributed in $\mathbb{G}$.

A Picture of Our Group.

- We have a bijection between $\mathbb{Z}_m$ and $\mathbb{G}$.
- f gives us an *efficient* (i.e. PPT) mapping from $\mathbb{Z}_m$ to $\mathbb{G}$. Specifically, given $x \in \mathbb{Z}_m$, we can compute $f(x)$ in $O(\log_2 m)$ group operations with the following algorithm: For each for $i \in [\lfloor \log_2 m \rfloor]$, compute $g^{2^i} := g^{2^{i-1}} \cdot g^{2^{i-1}}$. Then, let $x_1, \ldots, x_{|m|}$ be the bits of x, and we have

$$f(x) = \prod_{i \in [\lceil \log_2 m \rceil]} \left(g^{2^{i-1}} \right)^{x_i}$$

- There exists a reverse mapping $f^{-1} : \mathbb{G} \to \mathbb{Z}_m$. For $h \in \mathbb{G}$, $f^{-1}(h)$ finds $x \in \mathbb{Z}_m$ such that $h = g^x$. We call x the *discrete logarithm* of h.
- Is there an efficient way to compute f^{-1} in general? We don't know! Most people believe there is not.



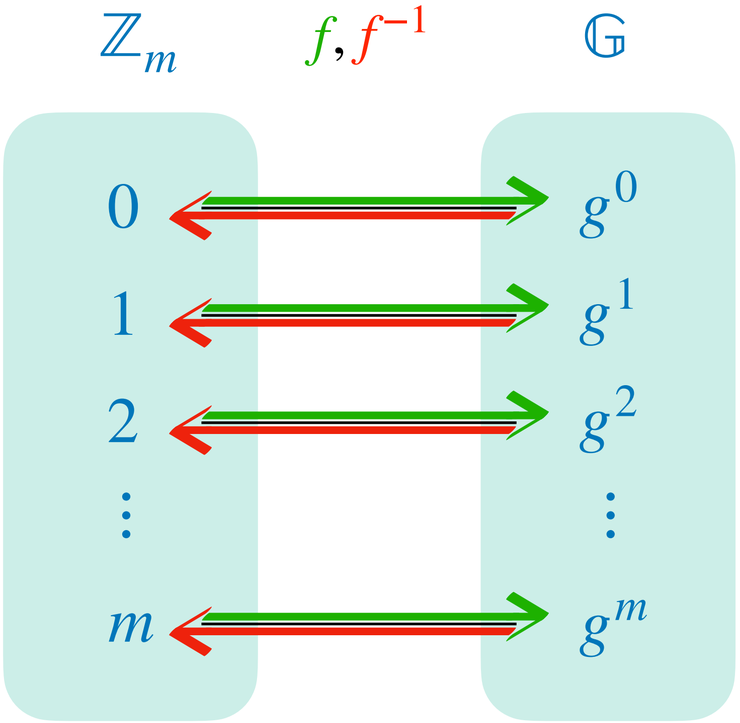
Try Your Hand at Discrete Logarithms

Example: Consider the cyclic group $(\mathbb{Z}_{13}^*, \cdot_{13})$ generated by 2.

- *What is the discrete logarithm of* 11 *with respect to* 2*?*
- *In other words, can you find* x *such that* $2^x \bmod 13 = 11$*?*
- Since $\langle 2 \rangle = \mathbb{Z}_{13}^*$, let's list the powers of 2:
  $\langle 2 \rangle = \{2, 4, 8, 3, 6, 12, 11, \ldots\}$
- So $\log_2(11) = 7$, but the way we found it seems inefficient.

Example: Consider the cyclic group $(\mathbb{Z}_{13}, +_{13})$ generated by 1.

- *What is the discrete logarithm of* 11 *with respect to* 1*?*
- *In other words, can you find* x *such that* $(x \cdot 1) \bmod 13 = 11$*?*
- Yes, it's 11. Finding discrete logs is easy in *some* groups.



Definition 10: The Discrete Logarithm Game

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.
- The *discrete logarithm game* $\mathsf{DLGame}_{\mathcal{G},\mathcal{A}}(\kappa)$ is played between a challenger and the adversary. The output of the game is the output of the challenger. The game is as follows:

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$h \leftarrow \mathbb{G}$

$(\mathbb{G}, g, q, h)$ →

← $x \in \mathbb{Z}_q$

outputs 1 if $g^x = h$
outputs 0 otherwise

The Discrete Logarithm Assumption

Definition 11 (Hardness for DLog): The Discrete Logarithm Problem is *hard* relative to $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$, there exists some negligible function ε such that

$$\Pr\left[\mathsf{DLGame}_{\mathcal{G},\mathcal{A}}(\kappa) = 1\right] \leq \varepsilon(\kappa)$$

Definition 12 (DLog Assumption): *The Discrete Logarithm Assumption* asserts that there exists some PPT algorithm $\mathcal{G}$ relative to which the discrete logarithm problem is hard.

Theorem 1: If the Discrete Logarithm Assumption is true, then *families* of one-way functions exist.

$\mathsf{DLGame}_{\mathcal{G},\mathcal{A}}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$h \leftarrow \mathbb{G}$

$(\mathbb{G}, g, q, h)$ →

← $x \in \mathbb{Z}_q$

outputs 1 if $g^x = h$
outputs 0 otherwise

Recap: One-Way Function Families

Definition 13 (OWF Family): a set of functions $\{f_i : \mathcal{X}_i \to \mathcal{Y}_i\}_{i \in I}$ indexed by some set I is a *one-way function family* if:

1. There exists a PPT algorithm $\mathsf{OWFGen}(1^\kappa)$ that outputs some $i \in I$.
2. $\forall i \in I, f_i$ is PPT.
3. $\forall$ PPT $\mathcal{A}$ $\exists$ a negligible ε such that $\forall \kappa \in \mathbb{N}$, $\Pr\left[\mathsf{OWFFGame}_{\mathsf{OWFGen},\mathcal{A}}(\kappa) = 1\right] \leq \varepsilon(\kappa)$

$\mathsf{OWFFGame}_{\mathsf{OWFGen},\mathcal{A}}(\kappa)$

$i \leftarrow \mathsf{OWFGen}(1^\kappa)$

$x \leftarrow \mathcal{X}_i$

$y := f_i(x)$

$1^\kappa, i, y$ →

← $x' \in \mathcal{X}_i$

outputs 1 if $f_i(x') = y$
outputs 0 otherwise

Constructing OWF Families

- The two games $\mathsf{DLGame}_{\mathscr{G},\mathscr{A}}(\kappa)$ and $\mathsf{OWFFGame}_{\mathsf{OWFGen},\mathscr{A}}(\kappa)$ look very similar. This is no accident!

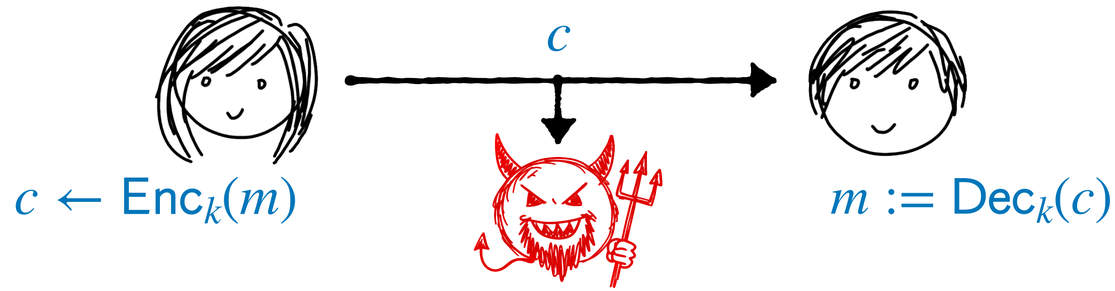
Theorem 1: If the discrete logarithm assumption is true, then one-way function families exist.

Proof Sketch:

- The sampling algorithm OWFGen is $\mathscr{G}$.
- We have $I = \text{image}(\mathscr{G})$, $\mathscr{X}_{(\mathbb{G},g,q)} = \mathbb{Z}_q$, and $\mathscr{Y}_{(\mathbb{G},g,q)} = \mathbb{G}$.
- $f_{(\mathbb{G},g,q)}(x) = g^x$.
- Any adversary that can invert $f_{(\mathbb{G},g,q)}$ with non-negligible probability must do so by finding the discrete logarithm of g^x for a uniformly sampled x, with non-negligible probability. Under the DL assumption, no such adversary can exist, and therefore, the family must be one-way. ∎

Revisiting A Problem from Lecture 4

- Now we have some kind of primitive that has a meaningful security property against bounded adversaries, but it's unclear exactly how the property of one-wayness can be useful to us.
- It turns out that OWFs imply symmetric encryption with keys that have length independent of the messages, and a lot else. It takes us the first half of CS6222 to prove this. We won't in CS4501.
- OWFs are *not known* to imply OT, so we're going to leave them behind for a little while. On our way to OT, we will investigate another protocol problem that we can model directly as a game (rather than using the real-ideal model), and solve directly using an assumption about groups.
- This will also let us see how assumptions can be *stronger* or *weaker* than one another.
- Recall that in Lecture 4 we introduced the notion of *symmetric encryption* using a *secret key* (we called it k). We assumed that Alice and Bob agreed beforehand on a *random* k that they know and the adversary doesn't. Today we'll find out how they can agree on k, even if there was never a time when they could communicate without the adversary listening.



Programming Techniques

S. L. Graham, R. L. ...
Editors

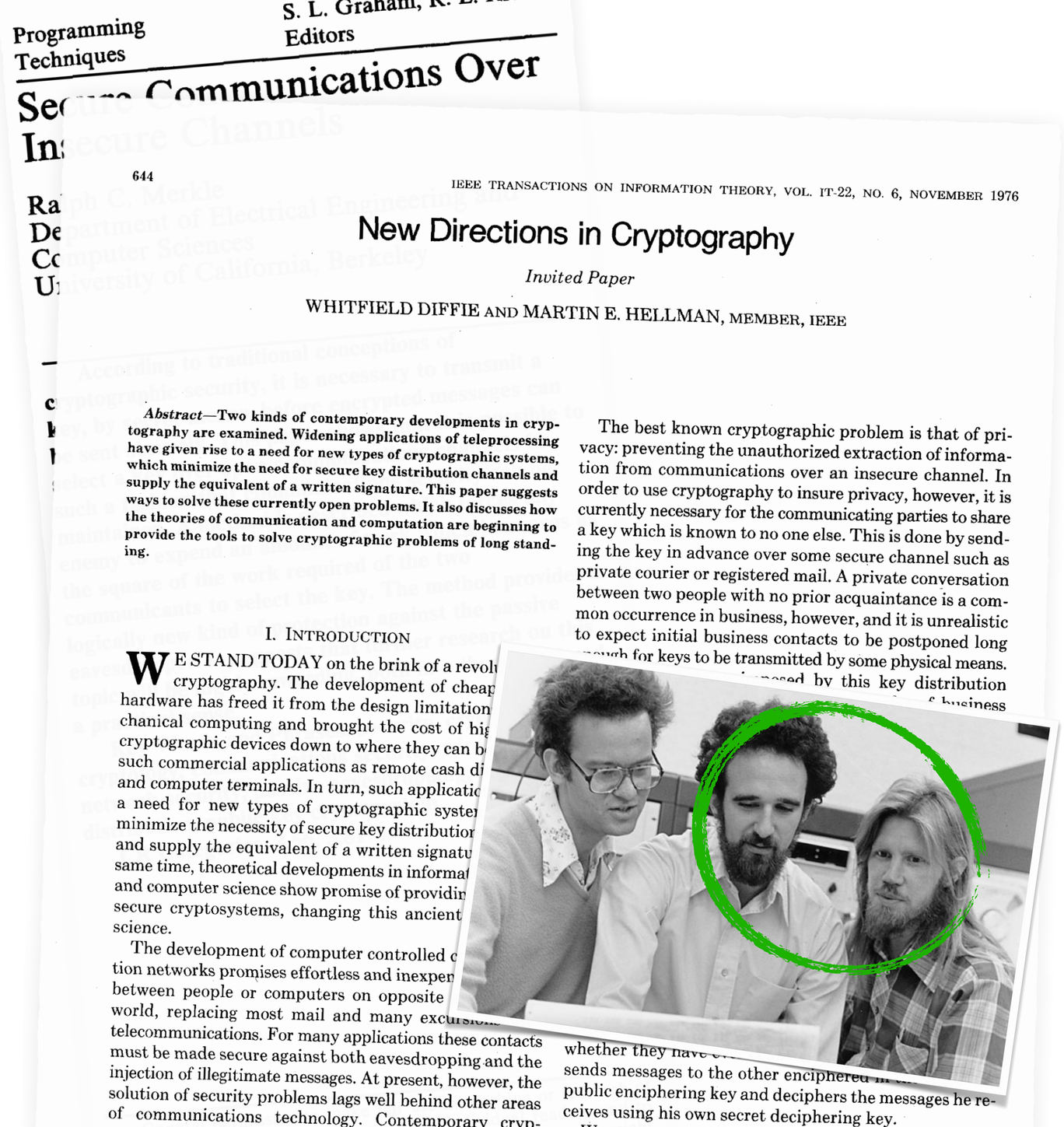
Secure Communications Over Insecure Channels

Ralph C. Merkle
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

According to traditional conceptions of cryptographic security, it is necessary to transmit a key, by secret means, before encrypted messages can be sent securely. This paper shows that it is possible to select a key over open communications channels in such a fashion that communications security can be maintained. A method is described which forces any enemy to expend an amount of work which increases as the square of the work required of the two communicants to select the key. The method provides a logically new kind of protection against the passive eavesdropper. It suggests that further research on this topic will be highly rewarding, both in a theo... a practical sense.

Key Words and Phrases: security, crypt... cryptology, communications security, wire... network security, passive eavesdropping, ... distribution, public key cryptosystem
CR Categories: 3.56, 3.81

- The problem of key agreement has been known since ancient times.
- In 1974, a PhD student called Ralph Merkle noticed that there is a way for two parties to agree on a secret key, such that even if the parties have no shared secrets to start with, and the adversary sees all communication between them, the adversary must work harder to find they key than the parties do.
- *Merkle's Puzzles* are an interactive process, rather than an algorithm that you apply to data (like encryption schemes are). This means it's probably the first *cryptographic protocol* ever.
- *Merkle's Puzzles* only make use of symmetric encryption, and Merkle proved that the adversary needs quadratically more time than honest parties to recover the key. Thus they provide very little security.
- Nevertheless, in 2008, Barak and Mahmoody (a UVa professor!) proved that Merkle's solution is the best possible solution if you only have symmetric encryption.
- "Further research will be highly rewarding"



644 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. Introduction

WE STAND TODAY on the brink of a revol
cryptography. The development of cheap
hardware has freed it from the design limitation
chanical computing and brought the cost of hig
cryptographic devices down to where they can b
such commercial applications as remote cash di
and computer terminals. In turn, such applicatic
a need for new types of cryptographic syste
minimize the necessity of secure key distributior
and supply the equivalent of a written signatu
same time, theoretical developments in informat
and computer science show promise of providin
secure cryptosystems, changing this ancient
science.

The development of computer controlled c
tion networks promises effortless and inexpen
between people or computers on opposite
world, replacing most mail and many excur
telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryp-

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. ...posed by this key distribution ...f business

whether they have ...
sends messages to the other enciphered ...
public enciphering key and deciphers the messages he receives using his own secret deciphering key.

- Merkle's advisor Martin Hellman was inspired by Merkle's idea. Together with Whitfield Diffie, Hellman formulated a precise mathematical assumption that would force the adversary to take *exponentially* more time than the honest parties to recover the key.
- Published in 1976, the *Diffie-Hellman Key Exchange* Protocol was the first *public-key* cryptosystem. This is often considered to be the beginning of *modern* cryptography.

- Diffie and Hellman received the Turing award for this work. They credit Merkle as a significant third contributor.

Def 14: Computational Diffie-Hellman Game

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.
- The *Computational Diffie-Hellman Game* $\mathsf{CDHGame}_{\mathcal{G},\mathcal{A}}(\kappa)$ is played between a challenger and the adversary. The output of the game is the output of the challenger. The game is as follows:

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$
$a, b \leftarrow \mathbb{Z}_q$
$(A, B) := (g^a, g^b)$

$(\mathbb{G}, g, q, A, B)$ →

← $C \in \mathbb{G}$

outputs 1 if $g^{a \cdot b} = C$
outputs 0 otherwise

The Computational Diffie-Hellman Assumption

Definition 15 (CDH Hardness): The Computational Diffie-Hellman Problem is *hard* relative to $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$, there exists some negligible function ε such that

$$\Pr\left[\mathsf{CDHGame}_{\mathcal{G},\mathcal{A}}(\kappa) = 1\right] \leq \varepsilon(\kappa)$$

Definition 16 (CDH Assumption): *The Computational Diffie-Hellman Assumption* asserts that there exists a PPT algorithm $\mathcal{G}$ relative to which the CDH problem is hard.

Think for a moment:
If the DL assumption is false, what do we know about the CDH assumption? Howabout if the DL assumption is true?

We say that *stronger* assumptions imply *weaker* ones.

$\mathsf{CDHGame}_{\mathcal{G},\mathcal{A}}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$
$a, b \leftarrow \mathbb{Z}_q$
$(A, B) := (g^a, g^b)$

$(\mathbb{G}, g, q, A, B)$ →

← $C \in \mathbb{G}$

outputs 1 if $g^{a \cdot b} = C$
outputs 0 otherwise

The Diffie-Hellman Key Exchange Protocol

- Let us assume that Alice (P_1) and Bob (P_2) are connected by an *authenticated channel* that is observed by a semi-honest adversary $\mathscr{A}'$ who does not corrupt either of them. That is, they can be certain that their messages are delivered faithfully to one another, but all messages can also be read by $\mathscr{A}'$.
- We cannot assume that Alice and Bob have any shared state to start. We wish them to *end* with a randomly-sampled secret k. Consider the following protocol:

$(\mathbb{G}, g, q) \leftarrow \mathscr{G}(1^\kappa)$
$a \leftarrow \mathbb{Z}_q,\ A := g^a$

$(\mathbb{G}, g, q, A)$

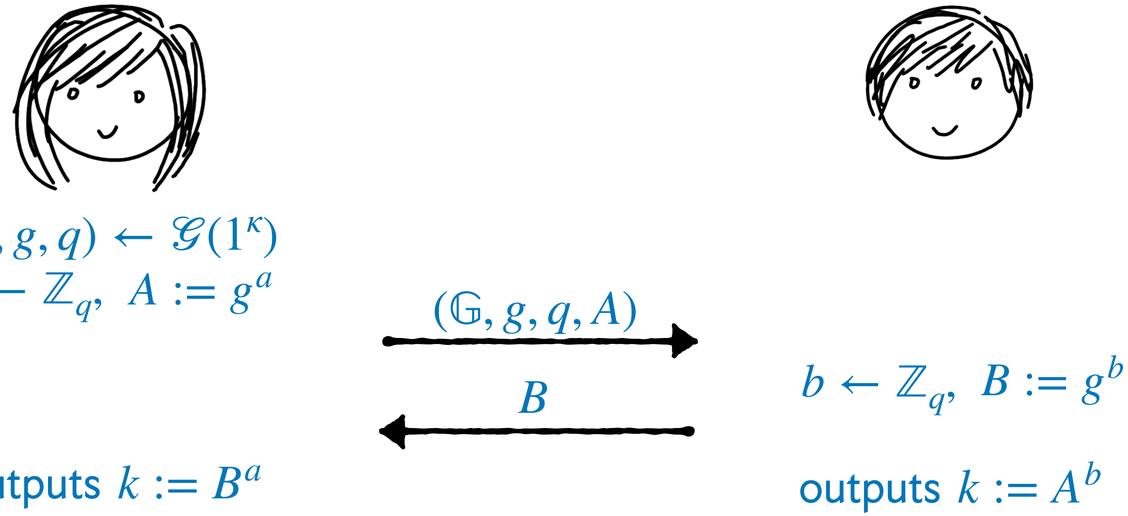
B

$b \leftarrow \mathbb{Z}_q,\ B := g^b$

outputs $k := B^a$

outputs $k := A^b$

The Diffie-Hellman Key Exchange Protocol



- During the execution of the DHKE protocol, $\mathcal{A}'$ witnesses the values $\mathbb{G}, g, q, A, B$. These values have exactly the same distributions as the values received by $\mathcal{A}$ in the CDH game!
- Question: *If the CDH assumption is true, what might that imply about the DHKE protocol?*
- Answer: Let T be a random variable representing the transcript in an instance of the DHKE protocol, and let K be a r.v. representing the output of P_1 and P_2. It seems intuitive to conclude that if the CDH assumption is true, then for every PPT $\mathcal{A}'$, there exists a negligible function ε such that $\Pr\left[\mathcal{A}'(1^\kappa, T) = K\right] \leq \varepsilon(\kappa)$. Don't worry yet exactly how the proof works.

The Diffie-Hellman Key Exchange Protocol

- Answer: Let T be a random variable representing the transcript in an instance of the DHKE protocol, and let K be a r.v. representing the output of P_1 and P_2. It seems intuitive to conclude that if the CDH assumption is true, then for every PPT $\mathscr{A}'$, there exists a negligible function ε such that $\Pr\left[\mathscr{A}'(1^\kappa, T) = K\right] \leq \varepsilon(\kappa)$. Don't worry yet exactly how the proof works.
- In other words, $\mathscr{A}'$ has a negligible probability of guessing the key that P_1 and P_2 agree upon.
- Think back to the one-time pad encryption scheme from Lecture 4. To send a message of length κ, OTP required a key k' that was uniform in $\{0,1\}^\kappa$. DHKE samples a key k that is uniform in $\mathbb{G}$ (which is itself sampled by $\mathscr{G}(1^\kappa)$). They're not quite compatible.
- For now, let's ignore the above problem and pretend that DHKE samples k' uniformly from $\{0,1\}^\kappa$. Resolving this for real requires a tool called a *randomness extractor*, which we won't cover in this class. If you want to learn about it, ask in office hours or join Grad Crypto!
- To prove OTP secure, we required a uniformly random, secret k'. Assuming the CDH assumption is true, our pretend version of DHKE provides a uniformly random k' that the adversary guesses with at-most negligible probability. *Is this good enough?*

The Diffie-Hellman Key Exchange Protocol

- To prove OTP secure, we required a uniformly random, secret k'. Assuming the CDH assumption is true, our pretend version of DHKE provides a uniformly random k' that the adversary guesses with at-most negligible probability. *Is this good enough?*
- What if the adversary learns half of the bits of k'? *Is OTP still secure given this knowledge?*
- **No!** If the adversary learns a particular bit of the key, then it can learn the corresponding bit of the message. So it can learn half of the message, which surely violates any notion of security you might expect an encryption scheme to have.
- *Can we rule out an adversary that learns half of the bits of k' if the CDH assumption is true?*
- **Not as far as we know!** It would be a breakthrough result if we figured out how! Of course, we're talking very imprecisely about a pretend version of DHKE that outputs bit-strings instead of group elements, but you could formulate precise statements of a similar flavor about the real DHKE protocol (think about the adversary learning whether C is in some *balanced subset* of $\mathbb{G}$). Proving them would be a surprise!

Toward Computational Indistinguishability

- In any case, we need to prove something stronger than $\Pr\left[\mathscr{A}'(1^\kappa, T) = K\right] \leq \varepsilon(\kappa)$, and it seems we'll need to start from an assumption that is stronger than CDH.
- We know that for the purposes of encryption, a *random* key would be good enough.
- Thinking back to the DHKE protocol, the adversary has complete information about the key, because it knows $A = g^a$ and $B = g^b$, and these imply a single, fixed value of $k = g^{a \cdot b}$.
- The CDH assumption told us that even though the adversary has *complete information* about k, it does not have enough computational power to *reconstruct* k completely.
- The stronger statement that we want to formalize is something like: "$k = g^{a \cdot b}$ is almost as good a random key in any application, even though it isn't random at all."
- When we talked about *perfect indistinguishability*, we relied upon the intuition that a distinguisher can't possibly tell two distributions apart if they are identical. It's also natural to say one distribution is "just as good" as another if no algorithm can distinguish them.
- Let's formalize this intuition about "just as good," and apply it to a *bounded* distinguisher.

Games for Distinguishers

- Remember: the distinguisher $\mathscr{D}$ is an algorithm. It's job is to receive a single sample from one of two distributions and determine which distribution that sample came from.
- We don't really care how $\mathscr{D}$ *labels* the two distributions. All we care is that it outputs something different on distribution 1, *relative* to what it outputs on distribution 0. Therefore we will pick two canonical labels—the string "1", and all other strings—and measure the *difference* between the probabilities that the outputs 1 given samples from each of the two distributions. Given a $\mathscr{D}'$ that uses any other pair of labels, we can build a $\mathscr{D}$ that uses these.
- We also don't care about individual samples, necessarily. If a particular sample from one of the distributions clearly indicates to $\mathscr{D}$ which of the two distributions it came from, but that sample almost never occurs, then we might still consider our distributions to be indistinguishable.
- In other words, we want to quantify how much better $\mathscr{D}$ is than a random guess, accounting for both the random coins that $\mathscr{D}$ might use and for the randomness of the two distributions.
- We call the measure of how much better $\mathscr{D}$ is than a random guess the *advantage* of $\mathscr{D}$.

Def 17: The Decisional Diffie-Hellman Game

- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q$, $|q| = \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (written multiplicatively). In other words, $\mathcal{G}$ samples a cyclic group given the security parameter.
- The *Decisional Diffie-Hellman Game* $\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},b}(\kappa)$ for $b \in \{0,1\}$ is played between a challenger and distinguisher. The game's output is the output of the distinguisher.

$\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a, b \leftarrow \mathbb{Z}_q$

$(A, B, C) := (g^a, g^b, g^{a \cdot b})$

$(\mathbb{G}, g, q, A, B, C)$ →

outputs a bit b'

$\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa)$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$a, b, c \leftarrow \mathbb{Z}_q$

$(A, B, C) := (g^a, g^b, g^c)$

$(\mathbb{G}, g, q, A, B, C)$ →

outputs a bit b'

The Decisional Diffie-Hellman Assumption

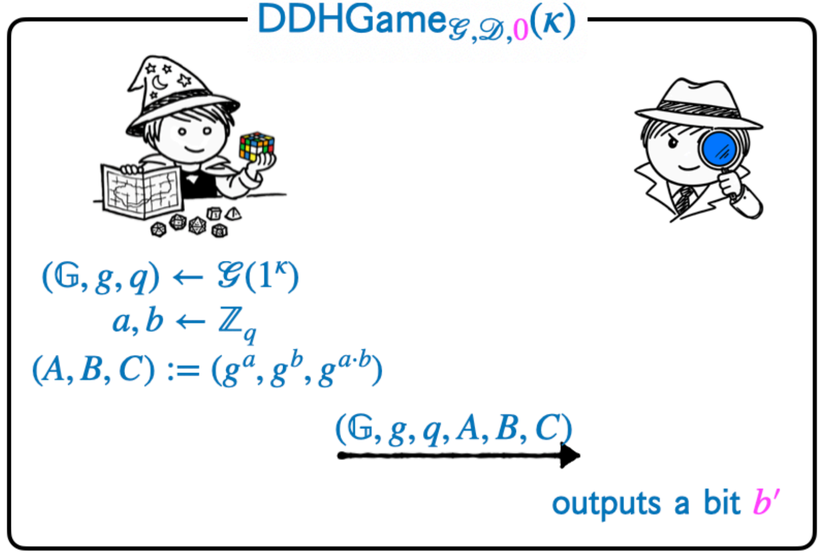
Definition 18 (DDH Hardness): The Decisional Diffie-Hellman Problem is *hard* relative to $\mathcal{G}$ if for every PPT distinguisher $\mathcal{D}$, there exists some negligible function ε such that

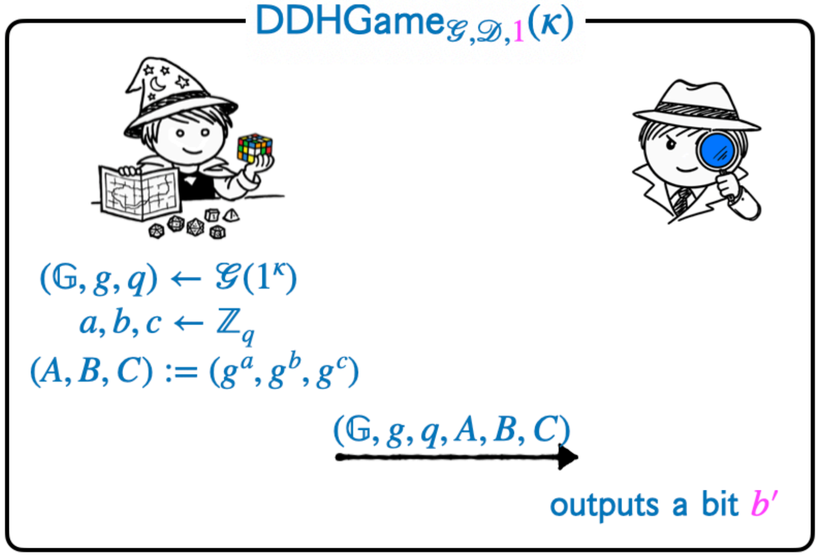
$$\left| \begin{array}{l} \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa) = 1\right] \\ \quad - \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa) = 1\right] \end{array} \right| \leq \varepsilon(\kappa)$$

Question: *Why can we never achieve advantage* $= 0$*?*
What easy strategy can $\mathcal{D}$ *always use?*

Definition 19 (DDH Assumption): *The Decisional Diffie-Hellman Assumption* asserts that there exists a PPT algorithm $\mathcal{G}$ relative to which the DDH problem is hard.

Question: *Is this stronger or weaker than CDH?*





Comparing Assumptions on a Group

	Discrete Log	Computational DH	Decisional DH
Type	Search ($\mathcal{A}$ solves a puzzle)	Search ($\mathcal{A}$ solves a puzzle)	Decision ($\mathcal{D}$ distinguishes distributions)
Intuition	Given g^x, computing x is almost as hard as guessing it.	Given g^a and g^b, computing $g^{a \cdot b}$ is almost as hard as computing a and b.	Given g^a, g^b, and g^c, determining whether $c = a \cdot b$ is almost as hard as computing $g^{a \cdot b}$.

DL assumption is false $\Longrightarrow$ CDH assumption is false $\Longrightarrow$ DDH assumption is false

DL assumption is true $\Longleftarrow$ CDH assumption is true $\Longleftarrow$ DDH assumption is true

Weaker $\longleftarrow$ Stronger

Toward Computational Security for Protocols

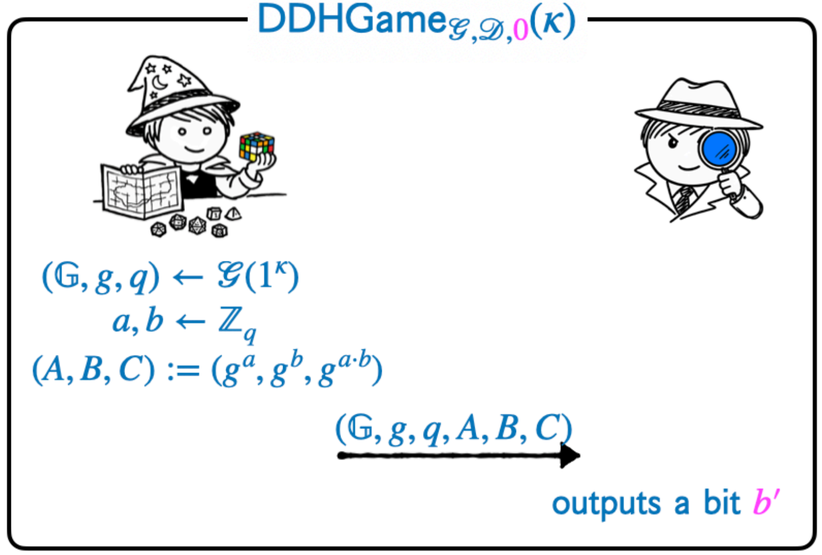
- Now we have an assumption that seems like an intuitive fit for the DHKE protocol. Since $g^{a \cdot b}$ is "almost as good as" g^c for some $c \leftarrow \mathbb{Z}_q$, even given knowledge of g^a, g^b, it seems natural to say that we can use $g^{a \cdot b}$ as a key wherever we could have used g^c.
- We prove the security of protocols in the real-ideal model, but we have some intuition about how it might work: in the real world, the parties send g^a, g^b and output $g^{a \cdot b}$, whereas in the ideal world, the functionality outputs g^c, and the simulator samples g^a, g^b randomly.
- **Problem:** We only believe that the DDH assumption can be true asymptotically. That is, a statement about a single value of κ makes no sense: we can only reason about an infinitely-increasing sequence of values of κ. So it seems that we will need an asymptotic notion of security for our protocol too!
- This is quite different from the *perfect security* notion that we have considered until this point, which was rooted in *perfect indistinguishability* without reference to a security parameter.
- We already arrived at a notion of *computational indistinguishability* for the Diffie-Hellman distribution specifically. Our next step will be to generalize this idea to other distributions.

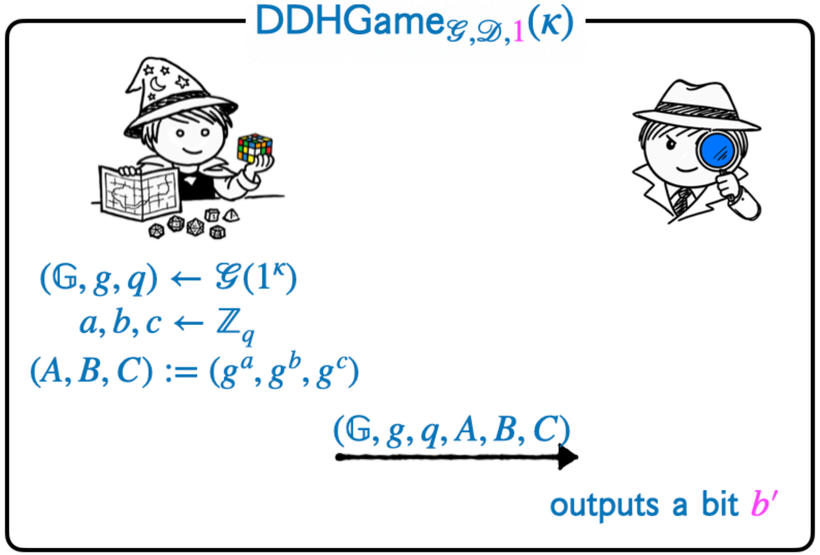
Recall: Decisional Diffie-Hellman

Definition 18 (DDH Hardness): The Decisional Diffie-Hellman Problem is *hard* relative to $\mathcal{G}$ if for every PPT distinguisher $\mathcal{D}$, there exists some negligible function ε such that

$$\left| \begin{array}{l} \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},0}(\kappa) = 1\right] \\ \quad - \Pr\left[\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},1}(\kappa) = 1\right] \end{array} \right| \leq \varepsilon(\kappa)$$

- Recall: For $b \in \{0,1\}$, the tuple $(\mathbb{G}, g, q, A, B, C)$ that is sent to $\mathcal{D}$ within $\mathsf{DDHGame}_{\mathcal{G},\mathcal{D},b}(\kappa)$ depends on $\kappa \in \mathbb{N}$.
- We are defining two *countably infinite* sequences of distributions (indexed by $\kappa \in \mathbb{N}$) and upper-bounding the difference in the outputs of $\mathcal{D}$, given samples from corresponding members of the sequences.
- We can make the same kind of statement about any arbitrary countably infinite sequences of distributions!





Computational Indistinguishability

Definition 20 (Ensemble): Suppose that for $\kappa \in \mathbb{N}$, X_κ is a distribution on some domain (these distributions and domains need not be related in any particular way). We say that the set $\mathfrak{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}} = \{X_1, X_2, \dots\}$ is an *ensemble of distributions.*

Def. 21 (Computational Indistinguishability): Let $\mathfrak{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}} = \{X_1, X_2, \dots\}$ and $\mathfrak{Y} = \{Y_\kappa\}_{\kappa\in\mathbb{N}} = \{Y_1, Y_2, \dots\}$ be ensembles. We say that $\mathfrak{X}$ and $\mathfrak{Y}$ *computationally indistinguishable* if, for every PPT distinguisher algorithm $\mathscr{D}$, there exists a negligible function ε such that

$$\left|\Pr\left[\mathscr{D}(1^\kappa, x) = 1 : x \leftarrow X_\kappa\right] - \Pr\left[\mathscr{D}(1^\kappa, y) = 1 : y \leftarrow Y_\kappa\right]\right| \leq \varepsilon(\kappa)$$

Notation 1: we use $\mathfrak{X} \approx_c \mathfrak{Y}$ to indicate that the ensembles $\mathfrak{X}$ and $\mathfrak{Y}$ are computationally indistinguishable.

Notation 2 : by abuse of notation, we use $\mathfrak{X} \equiv \mathfrak{Y}$ to indicate that the ensembles $\mathfrak{X}$ and $\mathfrak{Y}$ comprise distributions such that each corresponding pair are identical.

Recall: Simplified Definition for Semi-Honest $\mathcal{A}$

Definition 22. Perfect Semi-Honest SFE (Simplified):

Let $n, t \in \mathbb{N}$ such that $t < n$, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be a (possibly randomized) n-ary function, and let π be a n-party protocol.

We say that π *perfectly securely computes* f in the presence of a semi-honest adversary that statically corrupts up to t parties if there exists a *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every vector $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ we have

$$\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi)$$

Identically Distributed

Now: Simplified Definition for Bounded SH $\mathscr{A}$

Definition ~~22~~ 23. ~~Perfect~~ Computational Semi-Honest SFE (Simplified):

Let $\kappa, n, t \in \mathbb{N}$ such that $t < n$ and n is upper-bounded by some polynomial in κ, let $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$ be a (possibly randomized) n-ary function, and let π be a n-party protocol that runs in PPT relative to κ.

We say that π ~~*perfectly*~~ *securely computes* f in the presence of a bounded semi-honest adversary that statically corrupts up to t parties if there exists a PPT *simulator algorithm* Sim such that for every $I \subseteq [n]$ of size $|I| \leq t$ and every $\vec{x} \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ we have

$$\left\{ \left(\mathsf{Sim}\left(1^\kappa, I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x}) \right) \right\}_{\kappa \in \mathbb{N}} \approx_c \left\{ (\mathsf{VIEW}_I, \mathsf{OUTPUT}_\pi) \right\}_{\kappa \in \mathbb{N}}$$

Computationally Indistinguishable

Note: to make things simple, the parties will receive (the correct value of) 1^κ as input!

CS4501 ~~Cryptographic Protocols~~
Intro to Cryptography Speedrun
Lecture 13: Discrete Logarithms, Diffie-Hellman Key Exchange

https://jackdoerner.net/teaching/#2026/Spring/CS4501