Definitions

Security Definitions

$\mathcal{M}$ is the set of plaintext messages
$\mathcal{K}$ is the set of keys
$\mathcal{C}$ is the set of ciphertexts
Kg is a key generation algorithm (probabilistic and outputs a value in $\mathcal{K}$
E is an *enciphering algorithm*, **deterministically** enciphers an input message (from $\mathcal{M}$ *under* a key in $\mathcal{K}$ into a ciphertext in $\mathcal{C}$
D is a deciphering algorithm (the inverse of $+E$)

Kerckhoff's Principle

(roughly) when devising a security system, always assume that the adversary will know all the details of how the system works

Security Experiment

Specifies:

- adversary's goal
- how adversary can interact with the system

Enciphering Scheme

- A triple of algorithms Kg, E and D
  - Kg **randomly generates a key** $k \in K$
  - E takes a key k and message $m \in M$ to create a **ciphertext** $c \leftarrow E_k(m) \in C$
    - we are considering cases where $C = M$ i.e. ciphertexts are from the same space as plaintexts
  - D takes k and c and outputs a **purported** deciphered message $m' \leftarrow D_k(c)$
- So **an enciphering scheme is correct iff** for all k and m, $D_k(E_k(m)) = m$

Security Goals

- **One-wayness:** recovering the plaintext **in full** from the ciphertext should be hard
- **Perfect secrecy**: the ciphertext should reveal **no information** about the plaintext
  - This is captured by the notion that the distribution over ciphertexts induced by enciphering a random plaintext with a freshly generated key is **independent from the plaintext** (the plaintext doesn't affect the ciphertext distribution, so knowledge of the ciphertext doesn't narrow down the plaintext)
  - i.e. $\forall c \in C, m \in M, P\left(m* = m, m* \leftarrow_{\$} M | c* = c\right) = P\left(m* = m\right)$

- for all ciphertext-plaintext pairs, the probability that the a uniform-randomly selected plaintext $m* = m$ given that its enciphering $c* = c$ is the same as the unconditional probability $m* = m$
- so the condition that $c* = c$ doesn't affect the probability of m being any random plaintext $m*$

Properties

Commutativity

Commutativity is a property of an operation. $*$ is commutative, then $x * y = y * x$.

Associativity

Associativity is a property of an operation. $*$ is associative, then $x * (y * z) = (x * y) * z$

Injective

A function is injective if each output is mapped to by **at most one** input.

Surjective

A function is surjective if every output is mapped to by **at least one** input.

Bijection

A function is bijective if it is both injective and surjective - i.e. it is a perfect **one-to-one** mapping.

Group

A group $(S, +)$ is a combination of a set S and a binary operator $+$ that satisfies certain algebraic properties:

- existence of identity and inverses
- associativity

ElGamal Encryption

Setup

1. A chooses a prime p and an integer $n, 0 < n < p-1$
2. A chooses a random secret key $sk_a, 1 \leq sk_a \leq p-2$
3. A computes a public key $pk_a = n^{sk_a} \mod p$
4. A sends (p, n, pk_a) to B

Encryption

5. B chooses a random secret key $sk_b, 1 \leq sk_b \leq p-2$
6. B computes a public key $pk_b = n^{sk_b} \mod p$
7. B computes the shared secret $ss = pk_a^{sk_b} \mod p$
8. B encodes a message as an integer $\mod p$
9. B encrypts the message using $enc_m = m \times ss \mod p$
10. B sends (enc_m, pk_b) to A

Decryption

11. A computes the shared secret $ss = pk_b^{sk_a} \mod p$
12. A decrypts the message via $m = ss^{-1} \times enc_m \mod p$

Seven Pillars Diagnostic

1. a)
    - 29=16+8+4+1->11101
    - 33=32+1->100001
    - 110=64+32+8+4+2->1101110
    - 16->10000, 9->1001, ∴ 16x9=10010000
    b)
    - 11101000
    - 10110110
    - 11011001
    c) 64bit, ?
2. a)
   1.
   2. add items in increasing order of weight until you can't
   b) 37-28=9->1001, so first coin toss for (0,1,2,3,4) or (5,6,7,8,9), second coin toss for (0,1,2)

3. $2^{50} = (2^{10})^5 \approx (10^3)^5 \approx 10^{15}$ -> might be around limit of feasibility, $2^{64} \approx 10^{15} \times 2^{14}$ -> not feasible.
4.

Cryptology Lecture 3

Euclid's Algorithm

- If $d = gcd(a, b)$, then there exist integers m, n such that $am + bn = d$
  - If $a, b \in \mathbb{Z}, \gcd(a, b) = d$ then $\exists m, n \in \mathbb{Z} \mid am + bn = d$.
- Useful for finding inverses by setting $d = 1 \therefore am + bn = 1$
  - Only works if they have gcd of 1 (coprime)
- a is invertible $\mod n$ iff $\gcd(a, n) = 1$
- Find m, n using the iterative remainders approach
- Corrol

Groups

- A group is a pair of (set G, operation $*$) where $*$ maps a pair of elements in G to another element in G
- $(G, *)$ is a group (or "G defines a group under $*$") if the *group axioms* are satisfied:
  - It has an **identity**: $\exists e | \forall g \in G, g * e = e * g = g$
  - Every element has an **inverse**: $\forall g \in G, \exists h \mid g * h = h * g = e$
  - The operation is **associative**: $(a * b) * c = a * (b * c)$
- For any prime p, the set $\{1 \mod p, 2 \mod p, \ldots, (p-1) \mod p\}$ **is a group under multiplication**
- We call the set of integers mod n from 0 to n: $\mathbb{Z}/n\mathbb{Z}$
- and $(\mathbb{Z}/n\mathbb{Z})^*$ is the set of **invertible elements** in $\mathbb{Z}/n\mathbb{Z}$
  - if n is prime, then all elements are invertible under multiplication (so it forms a group)
- If the mod is a prime p, the group is *cyclic*: it can be *generated* using a number g as the sequence $g \mod p, g^2 \mod p, \ldots, g^{p-1} \mod p$
  - i.e. g generates $(\mathbb{Z}/p\mathbb{Z})^*$

  Definition 3.4. Let $(G, *)$ be a group. We say that $g \in G$ *generates* G if

  $$G = \{g, g * g, g \underbrace{* \cdots *}_{|G| \text{ times}} g\}.$$

  We then call g a *generator*.
-

Fermat's Little Theorem

- Euler ϕ function: count how many integers $0 < m < n$ are coprime with n
  - $\phi(p) = p - 1$ for p prime
  - $\phi(pq) = (p-1)(q-1)$ for q, p both prime
  -
- FLT: for every integer a and **squarefree** $n \in \mathbb{Z}_{>1}$ (squarefree = no square factors other than 1, i.e. no repeated prime factors)
  - $a^{\phi(n)+1} \equiv a \mod n$
  - if n is prime, then this means $a^{n-1} \equiv 1 \mod n$
    - because there are n-1 coprime integers (every integer between 1 (inclusive) and the prime (exclusive)), so $\phi(n) = n - 1$
    - and $a^{\phi(n)+1} \equiv a \mod n$ means $a^{\phi(n)} \equiv 1 \mod n$
    -

Sun-Tzu's Remainder Theorem

- n, m are coprimes > 1
- a, b are integers
- there exist c, d so that cm + dn = 1, and:
- x = bcm + adn mod mn uniquely satisfies x mod m = a and x mod n = b
- we might want to solve a problem like x = 2 mod 3, x = 2 mod 4
- this requires that the mods are coprime
- so we can find c, d using euclid's alg

Sets

- $\mathbb{N}$ - natural numbers. This includes all positive integers starting from 1. In some definitions, it also includes 0.
- $\mathbb{Z}$ - integers. It includes all positive and negative whole numbers, including 0.
- $\mathbb{Q}$ - rational numbers. These are numbers that can be expressed as a fraction of two integers, where the denominator is not zero.
- $\mathbb{R}$ - real numbers. This includes all rational and irrational numbers, but not complex numbers.
- $\mathbb{I}$ - irrational numbers. These are numbers that cannot be expressed as a simple fraction of two integers.
- $\mathbb{C}$ - complex numbers

Cryptology Lecture 7

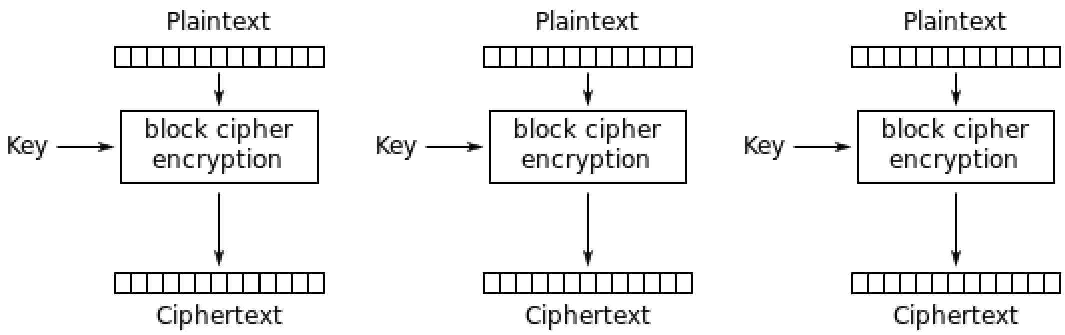
Baby-Step-Giant-Step

- Discrete log: we have a group $\mathbb{F}_p^*$, element g of order l (i.e. $g^l = 1$) and g^a, and we need to find a
- Alg:
    1. For i from 0 to $\sqrt{l}$, compute $b_i = g^i$
    2. For j from 0 to $\sqrt{l} + 1$, compute $c_j = g^a \cdot g^{-j\sqrt{l}}$, break if you find $c_j = b_i$
    3. Return $a = i + \sqrt{l} \cdot j$
- Costs at most $2\sqrt{l}$ multiplications, plus some setup costs (inversion and an exponentiation to $\sqrt{l}$'th power, these become negligible), so it is called $O(\sqrt{l})$

Index Calculus

Blockcipher Modes

Electronic Codebook (ECB)



Electronic Codebook (ECB) mode encryption

- **Insecure** because no **diffusion**: identical plaintext blocks have identical ciphertext blocks, revealing patterns

Nonce-Based Counter (CTR)

$\mathsf{Enc}_k^n(m)$

Assume block length ℓ and $m \in \{0,1\}^{\ell \cdot n}$, $\ell < 2^{\ell/2}$, $n \in \{0,1\}^{\ell/2}$

$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$
for $i \in \{1, \ldots, n\}$
  $X[i] \leftarrow n \| \langle i \rangle_{\ell/2}$
  $Y[i] \leftarrow \mathsf{E}_k(X[i])$
  $c[i] \leftarrow m[i] \oplus Y[i]$
$c \leftarrow c[1] \| \ldots \| c[n]$
return c

$\mathsf{Dec}_k^n(c)$

Assume block length ℓ and $c \in \{0,1\}^{\ell \cdot n}$, $\ell < 2^{\ell/2}$, $n \in \{0,1\}^{\ell/2}$

$(c[1], \ldots, c[n]) \leftarrow \mathsf{parse}(c)$
for $i \in \{1, \ldots, n\}$
  $X[i] \leftarrow n \| \langle i \rangle_{\ell/2}$
  $Y[i] \leftarrow \mathsf{E}_k(X[i])$
  $m[i] \leftarrow c[i] \oplus Y[i]$
$m \leftarrow m[1] \| \ldots \| m[n]$
return m

- Combine the nonce with a counter incremented for each block, and encipher the combination with k using the blockcipher
- Use the resulting set of pseudorandom ciphertexts as OTPs for the message blocks
- If the blockcipher used (E) is pseudorandom (indistinguishable), then CTR with E is nonce-based secure
-

Cipher Block Chaining (CBC)

$\mathsf{Enc}_k^n(m)$	$\mathsf{Dec}_k^n(c)$
Require $m \in \{0,1\}^{\ell \cdot n}$ and $n \in \{0,1\}^{\ell}$	Require $c \in \{0,1\}^{\ell \cdot n}$ and $n \in \{0,1\}^{\ell}$
$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$	$(c[1], \ldots, c[n]) \leftarrow \mathsf{parse}(c)$
$c[0] \leftarrow n$	$c[0] \leftarrow n$
for $i \in [1, \ldots, n]$	**for** $i \in [1, \ldots, n]$
$X[i] \leftarrow m[i] \oplus c[i-1]$	$X[i] \leftarrow \mathsf{D}_k(c[i])$
$c[i] \leftarrow \mathsf{E}_k(X[i])$	$m[i] \leftarrow c[i-1] \oplus X[i]$
$c \leftarrow c[1] \| \ldots \| c[n]$	$m \leftarrow m[1] \| \ldots \| m[n]$
return c	**return** m

- Insecure when nonces are reused

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

Diffie	Public	Hellman
generates secret key sk_D and public key pk_D		generates secret key sk_H and public key pk_H
	$pk_D \rightarrow, pk_H \leftarrow$	
generates **shared secret key** ss from sk_D and pk_H		generates **same** sk_S from sk_H and pk_D

- encode ss as a bit string $\in \{0,1\}^m$
- Diffie encrypts a message $m \in \{0,1\}^m$ as ciphertext $c = sk_S \oplus m$
- Diffie sends c to Hellman over a public channel
- Hellman decrypts c by xoring again - $m = c \oplus ss$

In more detail:

- a large prime p and a positive integer $n < p$ are chosen in public
- personal secret keys sk_D, sk_H are selected uniformly at random such that $0 < sk < p - 1$

- public keys are generated as $n^{sk} \mod p$
- shared secret key ss is generated as $ss = pk_H^{sk_D} = pk_D^{sk_H} = n^{sk_D sk_H} \mod p$
  - note that each party is able to generate the same value

Adversary's Game

Given $p, n, n^d \mod p, n^h \mod p$, find any of $d, h, n^{hd} \mod p$

- Discrete Logarithm Problem

ElGamal Encryption

Setup

Cryptology Lecture 2

Nonce-Based Encryption

- A nonce-based scheme E is a triple of algs (Kg, Enc, Dec)
  - Kg() -> k: key
  - Enc(k: key, n: nonce, m: message) -> c: ciphertext
  - Dec(n: nonce, c: ciphertext, k: key) -> m': message
- E is **correct iff** for all k that Kg can produce, all valid nonces and messages, the Dec of the Enc is m

Indistinguishability

- Consider a game:

$\mathsf{Exp}_E^{(n)\text{ind-real}}(\mathbb{A})$

$k^* \leftarrow_\$ \mathsf{Kg}$
$\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot)}$

$\mathcal{E}(n, m)$

no repeat nonces

$c \leftarrow \mathsf{Enc}_{k^*}^n(m)$
return c

$\mathsf{Exp}_E^{(n)\text{ind-ideal}}(\mathbb{A})$

$\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot)}$

$\mathcal{E}(n, m)$

no repeat nonces

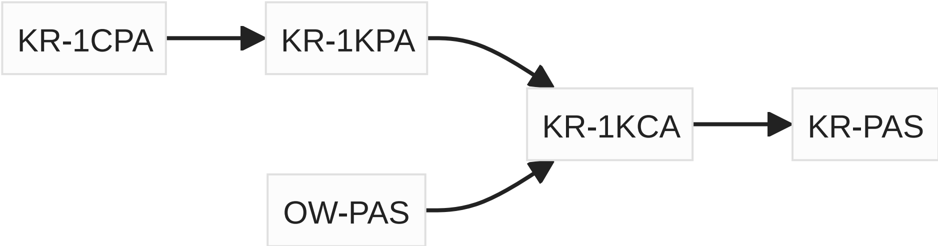
$c \leftarrow_\$ \mathcal{C}(|m|)$
return c

- The advantage of an attacker in distinguishing a nonce-based scheme from random ciphertexts is $$\text{Adv}*{E}^{(n)\text{ind}}(A) = \text{Pr}[\text{Exp}*{E}^{(n)\text{ind-real}}(A) : \hat{b} = 1] - \text{Pr}[\text{Exp}_{E}^{(n)\text{ind-ideal}}(A) : \hat{b} = 1]

$-note : therequirementfornorepeatmessageshasbeenchangedfornorepeatnonces, sot$

Reductions

From Strong to Weak Powers



- Security against "strong" adversaries (with more powers) should imply security against "weaker" adversaries (with fewer powers)
- We can make a reduction showing (t, ϵ)-KR-1CPA security implies (t, ϵ)-KR-1KPA
  - **If** the probability of A **recovering the key**, provided the ciphertext for 1 **chosen** plaintext and time $\leq t$ is $\leq \epsilon$
  - **Then** the scheme must also be (t, ϵ)-secure against an adversary who is provided a **known** (but not chosen) plaintext and ciphertext pair
- Strategy:
  - We want to show (t, ϵ)-KR-1CPA secure **implies** (t, ϵ)-KR-1KPA secure
  - -> Equivalent to (t, ϵ)-KR-1**K**PA **insecurity** implies (t, ϵ)-KR-1**C**PA **insecurity**
    - note, here we go from K->C
    - "if you're insecure against K, you **must** be insecure against C" is equivalent to "if you're secure against C, you're secure against K"
  - -> we want to show "If there's a working adversary against K, there also exists (/we can construct from it) an adversary against C"
  - We call the **new adversary** B the *reduction*
  - Two important claims about B:
    - B has same or better advantage
    - B runs in same or less time
  - These claims might be called the *analysis* of the reduction

$$\mathsf{Exp}_{E}^{\mathsf{kr\text{-}1cpa}}(\mathbb{B}_{\mathsf{kr\text{-}1cpa}})$$

$k^* \leftarrow_\$ \mathsf{Kg}$

$\widehat{k} \leftarrow_\$$

$$\mathbb{B}_{\mathsf{kr\text{-}1cpa}}^{\mathcal{E}_{\mathsf{cpa}}(\cdot)}$$

$\widehat{k} \leftarrow_\$ \mathbb{A}_{\mathsf{kr\text{-}1kpa}}^{\mathcal{E}_{\mathsf{kpa}}()}$

return $\widehat{k}$

$$\mathcal{E}_{\mathsf{kpa}}()$$

$m \leftarrow_\$ \mathcal{M}$

$c \leftarrow \mathcal{E}_{\mathsf{cpa}}(m)$

return (m, c)

$$\mathcal{E}_{\mathsf{cpa}}(m)$$

$c \leftarrow \mathsf{E}_{k^*}(m)$

return c

-
- The reduction $B_{kr-1cpa}$ (in outer dashed lines) just runs the adversary $A_{kr-1kpa}$ after sampling a plaintext randomly from the plaintext space. This sampling is its only overhead so it essentially runs in the same time as $A_{kr-1kpa}$.
- The two adversaries share a challenge key and key guess, so whenever A wins, so does B. This implies their advantage is the same.

From Hard to Easy Goals

- Just as **adversary capabilities** can be ranked, there is a similar hierarchy of **security goals**
- Usually this goes indistinguishability > one-wayness > key recovery
- A reduction from OW-PAS (passive one-wayness: no full recovery of the plaintext from the ciphertext) to KR-1KCA (key recovery from 1 known ciphertext):
  - My try:
    - "if E is OW-PAS secure, it is KR-1KCA secure" -> "if E is KR-1KCA insecure, it is OW-PAS insecure"
    - i.e. if an attacker can recover the key with 1 known ciphertext, a derived attacker can recover the full plaintext from a ciphertext

- essentially clear because if you can get the key from any (non-chosen) ciphertext, then you can get the plaintext using the decryption alg
- in other words, if you can't get the plaintext from the ciphertext, you **must** not be able to get the key from it (otherwise you would be able to get the plaintext)

-

Cryptographic Scheme Complexity

Cryptographic Scheme Complexity

The complexity of a λ-bit scheme is considered in terms of the number of **basic bit operations** required compared to λ.

Operations are considered easy if the number of basic bit operations is **polynomial** in λ.

Multiplication

We want to multiply $p \times q$.

Try the naive approach: add $\overbrace{q + \ldots + q}^{p \text{ times}}$
Since we have to perform p additions, and p has λ bits so is of the order of 2^λ, this algorithm has exponential complexity

Double-and-add

1. Write $p = \sum_{i=0}^{\lambda} c_i \times 2^i, c_i \in \{0, 1\}$ (write p into binary form)
2. **Compute:**

$$\begin{aligned} &2^0 \times q = q && \\ &2^1 \times q = q + q && \text{1 addition} \\ &2^2 \times q = 2^1 \times q + 2^1 \times q && \text{1 addition} \\ &\quad \ldots && \\ &2^\lambda \times q = 2^{\lambda-1} \times q + 2^{\lambda-1} \times q && \text{1 addition} \end{aligned}$$

   Total: λ additions
3. Observe that $pq = \left(\sum_{i=0}^{\lambda} c_i \times 2^i\right) \times q = \sum_{i=0}^{\lambda} c_i \times (2^i \times q)$
4. **Add:**

```
ans = 0
for i in range(lambda):
        if c_i = 1:
                ans += 2**i * q
```

This has $\le 2^\lambda$ additions

Exponentiation mod n

We want to exponentiate m^e.

Square-and-multiply

1. Write $e = \sum_{i=0}^{\lambda} c_i \times 2^i, c_i \in \{0,1\}$ (write e into binary form)
2. Observe $m^e = m^{\sum_{i=0}^{\lambda} c_i \times 2^i} = \prod_{i=0}^{\lambda} \left(m^{2^i}\right)^{c_i}$
3. **Square:**

$$\begin{aligned} m^{2^0} &= m \mod n && \text{0 multiplications} \\ m^{2^1} &= m^2 \mod n && \text{1 squaring} \\ m^{2^2} &= \left(m^2\right)^2 \mod n && \text{1 squaring} \\ \dots \\ m^{2^\lambda} &= \left(m^{2^{\lambda-1}}\right)^2 \mod n && \text{1 squaring} \end{aligned}$$

   Total: λ squarings
4. **Multiply:**

```
        ans = 0
        for i in range(lambda):
                if c_i = 1:
                        ans += (m ** (2 ** i)) % n
        ```

Total: $\le 2\lambda$ multiplications, $\therefore \le 4\lambda^2$
additions
```

Cryptology Lecture 5

- So far we haven't stopped adversaries from **modifying** messages
  - Most schemes we have looked at allow a predictable change in plaintext by modifying ciphertext (? length i guess)

# Message Authentication Codes

- We produce an auth tag from the key and message that can be used to verify they weren't modified since the tag was computed
- A MAC scheme looks like (Kg, Tag, Vfy) where:
  - Kg() -> k: key randomly generates a key
  - Tag(k: key, m: message) -> t: tag $\in T$
```

- Vfy(k: key, (m: message, t: tag)) -> valid: boolean
- MAC scheme is **correct** iff for all k and m: Vfy(k, (m, Tag(k, m))) = T
    - Note no bidirectional certainty - there can be collisions
- Usually Vfy just recomputes and compares the tag
- Some attacks:

$\mathsf{Exp}^{\text{euf-cma}}_{\text{MAC}}(\mathbb{A})$

$k \leftarrow_{\$} \mathsf{Kg}$
$(\widehat{m}, \widehat{t}) \leftarrow_{\$} \mathbb{A}^{\mathcal{T}_{\text{cma}}}$

$\mathcal{T}_{\text{cma}}(m)$

$t \leftarrow \mathsf{Tag}_k(m)$
return t

$\mathsf{Exp}^{\text{uuf-cma}}_{\text{MAC}}(\mathbb{A})$

$k \leftarrow_{\$} \mathsf{Kg}$
$m^* \leftarrow_{\$} \mathcal{M}$
$\widehat{t} \leftarrow_{\$} \mathbb{A}^{\mathcal{T}_{\text{cma}}}(m^*)$

$\mathcal{T}_{\text{cma}}(m)$

require $m \neq m^*$
$t \leftarrow \mathsf{Tag}_k(m)$ **return** t

Figure 5.1: Two different unforgeability experiments for MAC

- **Existential unforgeability under chosen message**
    - Easier for attacker
    - Attacker can get a tag for chosen messages, then has to try to tag a new one
    - Question is whether they can create **a** (i.e. at least one) valid (message, tag) pair, without the key
    - $$\mathrm{Adv}^{\text{euf-cma}}_{\text{MAC}}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\text{euf-cma}}_{\text{MAC}}(\mathcal{A}) : \mathrm{Vfy}_k(\tilde{m}, \tilde{t}) = \top \wedge \tilde{m} \text{ is fresh}]$$
    - The probability that the attacker produces a correct (m, t) pair, and they've never queried the oracle for the tag of that message before (fresh)
- **Universal unforgeability under chosen message**
    - Harder for attacker
    - Question is whether they can correctly tag **any/all message they are given**
    - $$\mathrm{Adv}^{\text{uuf-cma}}_{\text{MAC}}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\text{euf-cma}}_{\text{MAC}}(\mathcal{A}) : \mathrm{Vfy}_k(m^*, \tilde{t}) = \top]$$
    - ? so why is it "under chosen message"
        - because they can get tags for **chosen messages** from the oracle
- Existential freshness requires that the message $\hat{m}$ has not been queried to the oracle. Universal freshness requires that the message m^* does not get queried to the oracle.
- Existential forgery is a **weaker security goal** because the criteria for success are broader than for UUF. UUF-CMA security therefore implies EUF-CMA. We

can make a reduction from EUF to UUF.

CBC-MAC

$\text{CBC-MAC}_k(m)$

$(m[1], \ldots, m[n]) \leftarrow \mathsf{parse}(m)$
$X[0] \leftarrow 0^\ell$
for $i \in [1, \ldots, n]$
  $Y[i] \leftarrow X[i-1] \oplus m[i]$
  $X[i] \leftarrow \mathsf{E}_k(Y[i])$
return $X[n]$

$\text{C}^*\text{-MAC}_{k_1,k_2}(m)$

$(m[1], \ldots, m[n]) \leftarrow \mathsf{pad}(m)$
$X[0] \leftarrow 0^\ell$
for $i \in [1, \ldots, n]$
  $Y[i] \leftarrow X[i-1] \oplus m[i]$
  $X[i] \leftarrow \mathsf{E}_{k_1}(Y[i])$
$t \leftarrow \mathsf{F}_{k_2}(X[n])$
return t

Figure 5.2: CBC-MAC: the vanilla version for $\mathcal{M} = \{0,1\}^{\ell \cdot n}$ in the left panel; the usual template for dealing with $\mathcal{M} = \{0,1\}^*$ in the right panel.

- MAC based on using a blockcipher in CBC mode, retaining only the last block of ciphertext as the mac
- XOR each plaintext block with previous ciphertext block before being encrypted
  - So each ciphertext block depends on all plaintext blocks before it

Padding

- Want to turn a string of bits into a string of blocks, invertibly
- e.g. 10*: add a 1, then as many 0s as needed to hit block length

Hashes

- $H_k(m \in M) : d \in D$, and $|M| > |D|$ (**compression**)
- Often talking about **keyed** hash functions H_k
- Security notions
  - Collision resistance: probability (per try) that an attacker produces a pair of non-identical messages that have the same hash
  - Preimage resistance: probability (per try) attacker can produce a message which gives a hash chosen by us, given that hash
  - Second preimage resistance: probability (per try) attacker can produce a message with the same hash as a message chosen by us, given that message (and by extension able to determine the hash of it

Authenticated Encryption

- Nonce-Based Authenticated Encryption schem E = (Kg, Enc, Dec)
  - Kg generates key
  - Enc takes key, nonce and message and outputs ciphertext

> - Dec takes key, nonce and ciphertext and outputs plaintext **OR ⊥, representing decryption failure**
> - Correct iff Dec(k, n, Enc(k, n, m)) = m

$\mathrm{Exp}^{\text{ae-real}}_{\mathrm{ENC}}(\mathbb{A})$

$k \leftarrow_\$ \mathrm{Kg}$
$\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

$\mathcal{E}(n, m)$
no repeated nonces
$c \leftarrow \mathrm{Enc}^n_k(m)$
return c

$\mathcal{D}(n, c)$
c not output by $\mathcal{E}(n, \cdot)$
$m \leftarrow \mathrm{Dec}^n_k(c)$
return m

$\mathrm{Exp}^{\text{ae-ideal}}_{\mathrm{ENC}}(\mathbb{A})$

$k \leftarrow_\$ \mathrm{Kg}$
$\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$

$\mathcal{E}(n, m)$
no repeated nonces
$c \leftarrow_\$ \mathcal{C}(|m|)$
return c

$\mathcal{D}(n, c)$
c not output by $\mathcal{E}(n, \cdot)$
return $\perp$

-

> - AE-security
> - Attacker tries to distinguish between experiment with real AE oracles and one where ciphertexts are randomly generated and decryption always fails

(N)-IND-CCA

$\mathrm{Exp}^{(n)\text{ind-cca-real}}_{\mathrm{Enc}}(\mathbb{A})$	$\mathrm{Exp}^{(n)\text{ind-cca-ideal}}_{\mathrm{Enc}}(\mathbb{A})$
$k \leftarrow_\$ \mathsf{Kg}$ $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$	$k \leftarrow_\$ \mathsf{Kg}$ $\widehat{b} \leftarrow_\$ \mathbb{A}^{\mathcal{E}(\cdot,\cdot),\mathcal{D}(\cdot,\cdot)}$
$\mathcal{E}(n, m)$ no repeated nonces, and m not output by $\mathcal{D}(n, \cdot)$ $c \leftarrow \mathsf{Enc}^n_k(m)$ **return** c	$\mathcal{E}(n, m)$ no repeated nonces, and m not output by $\mathcal{D}(n, \cdot)$ $c \leftarrow_\$ \mathcal{C}(\lvert m \rvert)$ **return** c
$\mathcal{D}(n, c)$ c not output by $\mathcal{E}(n, \cdot)$ $m \leftarrow \mathsf{Dec}^n_k(c)$ **return** m	$\mathcal{D}(n, c)$ c not output by $\mathcal{E}(n, \cdot)$ $m \leftarrow \mathsf{Dec}^n_k(c)$ **return** m

- $\mathrm{Adv}^{(n)\text{ind-cca}}_{\mathrm{Enc}}(\mathcal{A}) = \Pr[\mathrm{Exp}^{(n)\text{ind-cca-real}}_{\mathrm{Enc}}(\mathcal{A}) : \hat{b} = 1] - \Pr[\mathrm{Exp}^{(n)\text{ind-cca-ideal}}_{\mathrm{Enc}}(\mathcal{A}) : \hat{b} = 1]$
-

Constructing AE

- Can construct an AE-secure scheme from a nonce-based indistinguishability-secure **encryption** scheme and an EUF-CMA-secure **MAC scheme**
- 3 typical ways to do this:

$\mathrm{MtE}^n_{k_a,k_e}(m)$	$\mathrm{EtM}^n_{k_a,k_e}(m)$	$\mathrm{E{+}M}^n_{k_a,k_e}(m)$
$t \leftarrow \mathsf{Tag}_{k_a}(n, m)$ $c \leftarrow \mathsf{Enc}_{k_e}(n, m \Vert t)$ **return** c	$c \leftarrow \mathsf{Enc}_{k_e}(n, m)$ $t \leftarrow \mathsf{Tag}_{k_a}(n, c)$ **return** $c \Vert t$	$c \leftarrow \mathsf{Enc}_{k_e}(n, m)$ $t \leftarrow \mathsf{Tag}_{k_a}(n, m)$ **return** $c \Vert t$

> - mac-then-encrypt, encrypt-then-mac, encrypt-and-mac

Cryptology Lecture 4

Diffie Hellman Key Exchange

- prime p and nonzero g are chosen publicly
- A and B each generate a keypair by choosing an integer d and determining $g^d \mod p$

- A and B trade their public keys g^d, g^h
- Then they raise the received public key to their own private key, so both end up with the shared secret $g^{dh} \mod p$

Security Parameter

- Some λ that is the key determining factor in a computation's runtime
- A computation might be "fast" e.g. polynomial in λ
  - i.e. you can define an upper bound for the number of basic operations needed as a polynomial in λ
- Or "slow" e.g. exponential or subexponential in λ
  - e.g. we might want the number of operations to be **lower**-bounded by $O(2^\lambda)$ (exponential) or $O(\lambda^\alpha \log_2(\lambda)^{1-\alpha}), \alpha \in (0,1)$ (subexpontial)
- We can increase λ to a size where polynomial algs are still fast (e.g. ms) and subexponential algs would take years
- In Diffie-Hellman, if $\lambda = \log_2(p)$, exponentiation $\mod p$ (for generation of keypair and computation of shared secret) should be polynomial in λ
  - But the inverse, computing the d^{th} roots $\mod p$ (the *discrete logarithm problem*), should be slower (subexponential)

ElGamal Encryption

- Setup:
  1. Diffie chooses a prime p and a generator g of $\mathbb{Z}/p\mathbb{Z} - \{0\}$.
  2. Diffie chooses a random secret $d \in \{1, \ldots, p-1\}$ and computes $pk_D = g^d \mod p$.
  3. Diffie sends his public key (p, g, pk_D) to Hellman.
- Encryption:
  1. Hellman chooses a random secret $h \in \{1, \ldots, p-1\}$ and computes $pk_H = g^h \mod p$.
  2. Hellman computes the shared secret $ss = pk_D^h \mod p$.
  3. Hellman computes the encrypted message $enc_m = m \cdot ss$.
  4. Hellman sends the ciphertext (pk_H, enc_m) to Diffie.
- Decryption:
  1. Diffie computes the shared secret $ss = pk_H^d \mod p$.
  2. Diffie computes the ciphertext $m = enc_m \cdot ss^{-1} \mod p$, which simplifies to $m = enc_m \cdot pk_H^{p-1-d} \mod p$.
- Note:
  - The inverse of the shared secret, ss^{-1}, is $g^{h^{p-1-d}}$ i.e. pk_H^{p-1-d}
  - If an attacker has both plaintext and ciphertext, they can recover the shared secret ss.

- So a new random per-message secret h should be used for each message

RSA

- Keygen:
    1. Pick two λ-bit primes $p \neq q$
    2. Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
    3. Pick e coprime to $\phi(n)$
    4. Compute $d = e^{-1} \mod \phi(n)$, the inverse of e
    5. Keypair $pk, sk = (e, n), (d, n)$
- Encrypt:
    - Message is $m \in \mathbb{Z}_{[0,n-1]}$
    - Ciphertext is $c = m^e \mod n$
- Decrypt:
    - $m = c^d \mod n$
- #todo: more understanding
- We want:
    - Fast keygen steps 2 and 4 i.e. fast multiplication and fast inverse finding
    - Attacker not to be able to find d from (e, n)
        - Since d can be found quickly as $e^{-1} \mod \phi(n)$, we need it to be slow to find $\phi(n)$ from n, as n is public
        - Since $n = pq$, finding $\phi(n)$ basically involves determining n's two prime factors
    - Encrypt should be fast (exponentiation $\mod n$)
    - Recovering m from c to be hard
        - This could theoretically be done since e, n are public, and $c = m^e \mod n \therefore m = c^{1/e} \mod n$
        - So we need computing e'th roots $\mod n$ to be hard/slow

Fast Multiplication: Double-and-Add

- To multiply pq
- Write q in binary as $\sum_{i=0}^{\lambda} q_i \times 2^i$
- So $pq = \sum_{i=0}^{\lambda} q_i \times 2^i \times p$
- Compute the $2^i \times p$ terms by simply repeatedly doubling p - only one addition per iteration
    - For a λ-bit number, this costs λ additions (polynomial)
- Then just iterate the bits of q and sum the $2^i \times p$ terms for which $q_i = 1$
    - This costs at most λ additions
- So overall at most 2λ additions

Fast Exponentiation: Square-and-Multiply

- $m^e = \prod_{i=0}^{\lambda} (m^{2^i})^{e_i}$
- Product of those m^{2^i} terms where e_i is 1
- We can make m^{2^i} by repeatedly squaring m, each time takes 1 multiplication i.e. 2λ basic operations, and there are λ squarings so overall $2\lambda^2$ basic ops
  - note the $\mod n$ here is important as otherwise you'd run out of memory fast
- Taking the product of those terms takes at most λ multiplications, depending on the number of 1 bits in e
  - So at most $2\lambda^2$ basic ops
- So square-and-multipy takes $\leq 4\lambda^2$ ops

Discrete Log Problem

- Given $g \mod p, g^d \mod p$, find $d \in [0, p-1]$ |
- We want this to be hard
- In Diffie-Hellman, we usually choose a g which is a generator of the group $(\mathbb{Z}/p\mathbb{Z})^*$*
  - This means that private keys $g^d \mod p$ take $p-1$ different values, i.e. there are no collisions where $g^d \mod p$ is the same for two different values of d with $1 \leq d \leq p-1$
  - So the key space is maximised: the private key d can correspond to any value in the group, avoiding reducing entropy

Order

- The *order* of an **element** is the minimum d such that g^d (or $g * g * \ldots * g$ d times) $= id$ (the identity of the group
  - A generator necessarily has order $p-1$ because if it reached the identity before $p-1$ elements it would repeat itself, missing some elements
- If g is a generator and l divides $p-1$, then $g^{\frac{p-1}{l}} \mod p$ has order l
  - if $p-1 = kl$, $g^{\frac{p-1}{l}} = g^{\frac{kl}{l}} = g^k$
  - order of $g^k = \frac{p-1}{k} = l$
- Problem: find int a such that $2^a = 17 \mod 37$. We are told 2 is a generator of the group.

- [ ] #todo : understand end of WS 4
- [ ]

Cryptology Lecture 1

Learning Schedule

One-Time Ciphers

Perfect Secrecy

Considering schemes where $M = C$

Key Recovery

Key recovery security experiment:

$$Exp_E^{kr-pas}(\overbrace{A}^{\text{adversary}})$$

$$k \leftarrow Kg$$

$$\hat{k} \leftarrow A()$$

A wins if $\hat{k} = k$ (the guessed key is correct)

- The **advantage** of an adversary in passively recovering the key is defined as the probability that their guess is correct:

$$\overbrace{Adv_E^{kr-pas}(A)}^{\text{advantage of A}} = Pr[Exp_E^{kr-pas}(A) : \hat{k} = k]$$

- We say that E is $(t, \epsilon) - kr - pas$ secure if for any adversary A that runs in time $\leq t$, the *advantage* of A is bounded by ϵ.
- If the adversary has no information about the system, the best they can do is guess the key:

$$\begin{aligned} &A_{\text{guess}}() \\ &\qquad k \leftarrow Kg \\ &\qquad returnk \end{aligned}$$

- In this case, the running time of the adversary $t = t_{Kg}$, and the advantage $\epsilon = \frac{1}{|\mathcal{K}|}$ (one over the size of the keyspace).

Bits of Security

- currently, 80 bits is considered enough for security
- 128 bits is considered secure for next 10 years
- 256 bits is recommended

One-Time Known Ciphertext Attack

Key is generated, adversary is given the scheme and one generated ciphertext

$$\begin{aligned} & Exp_E^{\text{kr-1kca}}(A) \\ & \quad k \leftarrow Kg \\ & \quad m \leftarrow \mathcal{M} \\ & \quad c \leftarrow E_k(m) \\ & \quad \hat{k} \leftarrow A(c) \end{aligned}$$

One-Time Known Plaintext Attack

Key is generated, adversary is given the scheme, one plaintext and its corresponding ciphertext

$$\begin{aligned} & Exp_E^{\text{kr-1kca}}(A) \\ & \quad k \leftarrow Kg \\ & \quad m \leftarrow \mathcal{M} \\ & \quad c \leftarrow E_k(m) \\ & \quad \hat{k} \leftarrow A(m, c) \end{aligned}$$

Theorems

- OTP satisfies perfect security
- Shannon's Theorem:
  - Scheme $E = (Kg, E, D)$ is only **perfectly secure** iff Kg draws from K uniformly at random, and for all (m,c) pairs there is exactly one unique key k such that $E_k(m) = c$
  - OTP is the **only enciphering scheme** with perfect security
- An enciphering scheme has perfect secrecy if and only if it has perfect **indistinguishability**
-

Indistinguishability

- A scheme E satisfies perfect indistinguishability iff for all c and m, the probability that m enciphers to c with a random key k is $1/|C|$, i.e. the encipherings are evenly distributed over the ciphertext space.
  - $\forall c \in C, m \in M,\ \Pr[c^* = c | m^* = m] = |C|^{-1}$
  - Given a particular message m *and its resulting ciphertext c* (when encrypted with a randomly selected key from K), the distribution of c* over C is uniformly random, so the probability that it equals some particular ciphertext c is $1/|C|$
  - No particular ciphertext is more likely than any other, making it impossible for an observer to gain any information about the plaintext by looking at the ciphertext
- In game-based terms, we can consider a game where the attacker tries to distinguish two different experiments:

$$\begin{array}{l} \mathsf{Exp}_E^{\text{1ind-real}}(\mathbb{A}) \\ \hline k \leftarrow_\$ \mathsf{Kg} \\ m \leftarrow_\$ \mathbb{A} \\ c^* \leftarrow \mathsf{E}_k(m) \\ \widehat{b} \leftarrow_\$ \mathbb{A}(c^*) \end{array} \qquad \begin{array}{l} \mathsf{Exp}_E^{\text{1ind-ideal}}(\mathbb{A}) \\ \hline k \leftarrow_\$ \mathsf{Kg} \\ m \leftarrow_\$ \mathbb{A} \\ c^* \leftarrow_\$ \mathcal{C} \\ \widehat{b} \leftarrow_\$ \mathbb{A}(c^*) \end{array}$$

- in each game, a key is randomly chosen and the attacker provides a message m.
- in one game, the ciphertext is produced by encrypting m, and in the other, it is selected randomly from C
- then the attacker must try to decide whether the ciphertext is correct (i.e. returning some boolean $\hat{b}$ when provided $c*$)
- The advantage of the attacker in *one-time distinguishing E from random* is $$\text{Adv}*{E}^{\text{ind}}(A) = \text{Pr}[\text{Exp}*{E}^{\text{ind-real}}(A) : \hat{b} = 1] - \text{Pr}[\text{Exp}_{E}^{\text{ind-ideal}}(A) : \hat{b} = 1]

$-i.e. thedifferenceinprobabilitiesthatAwillreturn1foreachexperiment - Ifthediffe$

Blockciphers

- A blockcipher with block length l is a **symmetric enciphering scheme** with $M = C = \{0, 1\}^l$
  - i.e. the message and cipher space are all permutations of l binary bits
- A blockcipher is effectively a *permutation* as it maps every possible plaintext block onto a unique ciphertext block, and $M = C$ so it's effectively a permutation, (a reordering, a mapping)

Key Recovery

- Consider a game where we generate a key and then the adversary can repeatedly request encryption of a plaintext with that key and receive the ciphertext

```
Exp_E^{kr-cpa}(𝔸)
k* ←$ Kg
k̂ ←$ 𝔸^{ℰ(·)}

  ℰ(m)
  c ← E_{k*}(m)
  return c
```

```
𝔸_search
Ks ← ∅
for k ∈ 𝒦
   if E_k(m) = c
   then Ks ← Ks ∪ {k}
k̂ ←$ Ks
return k̂
```

- They can brute-force (exhaustively search) the key space:
  - Choose some random plaintext m
  - Iterate through all $\hat{k} \in K$:
    - Encipher m with $\hat{k}$ to get $\hat{c}$
    - Request enciphering of m from us to receive c
    - Compare c with $\hat{c}$, if they are equal the key matches ours
- In this context the advantage is

$$\mathrm{Adv}_E^{\text{kr-cpa}}(A) = \Pr[\mathrm{Exp}_E^{\text{kr-cpa}}(A) : \hat{k} = k^*]$$

  - The probability that the adversary guesses the correct key
  - We say E is (t, q, ϵ)-secure if the advantage is $\leq \epsilon$ for an adversary running in time at most t and with at most q queries from the *oracle* for encipherings

Indistinguishability (Pseudorandomness)

- With block ciphers, indistinguishability is called pseudorandomness
  - I guess like "fake-random", like you can't tell if it's random, rather than it's trying to actually be random

```
Exp_E^{ind-real}(𝔸)
k* ←$ Kg
b̂ ←$ 𝔸^{ℰ(·)}

  ℰ(m)
  no repeat queries
  c ← E_{k*}(m)
  return c
```

```
Exp_E^{ind-ideal}(𝔸)

b̂ ←$ 𝔸^{ℰ(·)}

  ℰ(m)
  no repeat queries
  c ←$ 𝒞
  return c
```

-
- In this game, the attacker tries to distinguish between the two experiments

- Crucially, the attacker **cannot make repeat queries** with the same input as then they would observe that one experiment produces different c for the same m (as it's in fact random, rather than real enciphering), giving the game up
- The advantage is:

$$\mathrm{Adv}_E^{\text{ind}}(A) = \Pr[\mathrm{Exp}_E^{\text{ind-real}}(A) : \hat{b} = 1] - \Pr[\mathrm{Exp}_E^{\text{ind-ideal}}(A) : \hat{b} = 1]$$

> - The diff in probability that it gives 1 for the correct experiment vs the wrong one

Birthday Bound

- An adversary that makes q queries to a random permutation (the "fake" experiment) will find a collision with probability apprx $\frac{q \times (q-1)}{2 \times |C|}$
- This "birthday bound" places a constraint on the block length l of the blockcipher, requiring it be above some minimum level to reduce collision probability to a desirable level
- Birthday bound reverse: $n = \sqrt{2N \ln(\frac{1}{1-p})}$

RSA

RSA

- A **message encryption scheme**

Euler φ Function

For a positive integer n, $\varphi(n) = |\{a \in \mathbb{Z} | 1 \leq a < n and \gcd(a, n) = 1\}|$
For example: $\varphi(5) = |\{1, 2, 3, 4\}| = 4$, $\varphi(6) = |\{1, 5\}| = 2$

For two primes p, q, $\varphi(pq) = (p-1)(q-1)$

Setup (keygen)

1. A picks two λ-bit primes $p \neq q$
2. A computes $n = p \times q$ and $\varphi(n) = (p-1)(q-1)$
3. A chooses e coprime to $\varphi(n)$
4. A computes $d \equiv e^{-1} \mod \varphi(n)$
5. A generates a keypair: $pk_A = (e, n), sk_A = (d, \varphi(n))$
6. A sends pk_A to B

Encryption

7. B encodes a message as $m = \{0, \dots, n-1\}$
8. B computes $enc_m = m^e \mod n$
9. B sends enc_m to A

Decryption

10. A computes $m = {enc_m}^d \mod n$
    - $= m^{ed} \mod n = m^{1+k\varphi(n)} = m \times {m^{\varphi(n)}}^k \mod n$

```
Error parsing Mermaid diagram!

Parse error on line 2:
...uenceDiagram      p
-----------------------^
Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW',
'DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT',
'DOTTED_POINT', got 'NEWLINE'
```

Fermat's Little Theorem

Let $p \neq q$ be primes, and $a \in \mathbb{Z}$ such that $\gcd(a, pq) = 1$ (a is *coprime* to pq).

$$a^{p-1} \equiv 1 \mod p$$

$$a^{(p-1)(q-1)} \equiv 1 \mod pq$$

$$a^{-1} \equiv a^{p-2} \mod p$$

Adversary's Game

Given $e, n, m^e \mod n$, adversary wants to find **any** of $d, \varphi(n), m$.

- Factoring n to find p, q will allow finding $\varphi(n) = (p-1)(q-1)$ (see Euler's Phi Function)
- Finding m from $e, n, m^e \mod n$ is related to the Discrete Logarithm Problem

Efficiency

Efficient operations needed:

- picking primes
- multiplying
- choosing coprimes
- inversion mod $\varphi(n)$
- exponentiation $\mod n$

Notes

Things we want to be hard:

- $n \to \varphi(n)$ i.e. factoring
- $c \to m$ i.e. computing e^{th} roots $\mod n$
-

Digital Signatures

ElGamal Signatures

- Choose prime p, generator g
- Choose a random private key $0 < a < p-1$ and make public key $pk = g^a \mod p$
- Choose a message $\mod p-1$ (because the message is in the **exponent** of g)
- Sign:
  - Choose a nonce $0 < k < p-2$, coprime to $p-1$
  - Compute $r = g^k \mod p$
  - Compute $sig = k^{-1}(m - ar) \mod (p-1)$
  - Publish (r, sig)
- Verify:
  - Check $g^m = pk^r \cdot r^{sig} \mod p$
    - because $pk^r \cdot r^{sig} = g^{ar} \cdot g^{k \cdot k^{-1}(m-ar)}$
    - $= g^{ar+m-ar} = g^m$
- Nonce must be kept secret to avoid key recovery (m, $r = g^k$, $sig = k^{-1}(m-ar)$ are all public -> $k \cdot sig = k \cdot k^{-1}(m-ar) = m - ar$ -> $a = \frac{m-k \cdot sig}{r}$
- Nonce must not be reused: allows attacker to recover k and therefore a

```
_
```

Cryptology Lecture 6

Pohlig-Hellman

- Use SRT to solve some types of discrete log problem:
- for a prime p a generator g, and a group member $x = g^a (\mod p)$, find a
- FLT: $a^{\phi(n)+1} = a \mod n$, and for primes $a^p = a \mod p$ (and so $a^{p-1} = 1 \mod p$)
  - So $g^{a+(p-1)k} = g^a \mod p$ for any integer k ($k \in \mathbb{Z}$)

1. We factorise $\phi(p) = p - 1$ into prime powers: $p - 1 = q_1^{e_1} \cdot q_2^{e_2} \cdot \ldots \cdot q_r^{e_r}$, where q_i are prime

1. Choose one of those factors e.g. q_r, and raise x to the power of $\phi(p)/q_r$:

$$x^{\phi(p)/q_r} = (g^a)^{\phi(p)/q_r}$$

2. Divide a by q_r and express it as a quotient c and remainder d:

$$\begin{aligned} x^{\phi(p)/q_r} &= (g^a)^{\phi(p)/q_r} \\ &= (g^{cq_r+d})^{\phi(p)/q_r} \\ &= (g^{cq_r})^{\phi(p)/q_r} \cdot (g^d)^{\phi(p)/q_r} \\ &= (g^{\phi(p)})^q \cdot g^{d\phi(p)/q_r} \\ &= g^{d\phi(p)/q_r} \end{aligned}$$

4. $g^{d \cdot \phi(p)/q_r}$ is g t

3. For each q_i:
   - Write $a = a_0 + a_1 q_i + a_2 q_i^2 + \ldots$, with $a_j \in [0, q_i - 1]$
   -

- Then we can find $a \mod (p-1)$ from all the $a \mod q_i^{e_i}$s
-