proa

lunedì 10 gennaio 2022 14:52

proa

Table of Contents

Solutions to Information Security exams

This is a just a collection of exam solutions made by a group of students to help in the future. Probably many solutions contain errors and are not complete. Please correct the wrong ones, add solutions of future exams in this jupyter notebook and **share the knowledge**.

Don't sell this document, please. The intended usage is just for students of the Information Security course at University of Padova, which already can access to the text of past exams. Any other usage is strictly forbidden.

Thank you.

A group of students.

Communication exercises

Possible attacks: man in the middle (which can be mitigated with digital signature), reply attack (like danning sacco). If sign then encrypt is used it might be vulnerable to padding oracle attack.

Mockup 2020 and 2021

Consider the following (naïve) entity authentication protocol.

A wants to authenticate himself to B, with whom he shares a secret key $k \in \{0,1\}^\ell$. B employs the following N-round protocol, which does not require A to send k over the channel.

For $n = 1$ to N

$n.1$ B : generates a challenge $x_n \sim \mathcal{U}(\{0,1\}^\ell)$

B → A : x_n

$n.2$ A : calculates $y_n = x_n \oplus k$ and $b_n = \begin{cases} 1 & \text{if } y_n \text{ has an even number of ones} \\ 0 & \text{if } y_n \text{ has an odd number of ones} \end{cases}$

A → B : b_n

$n.3$ B : checks b_n. If b_n is incorrect, the authentication is rejected

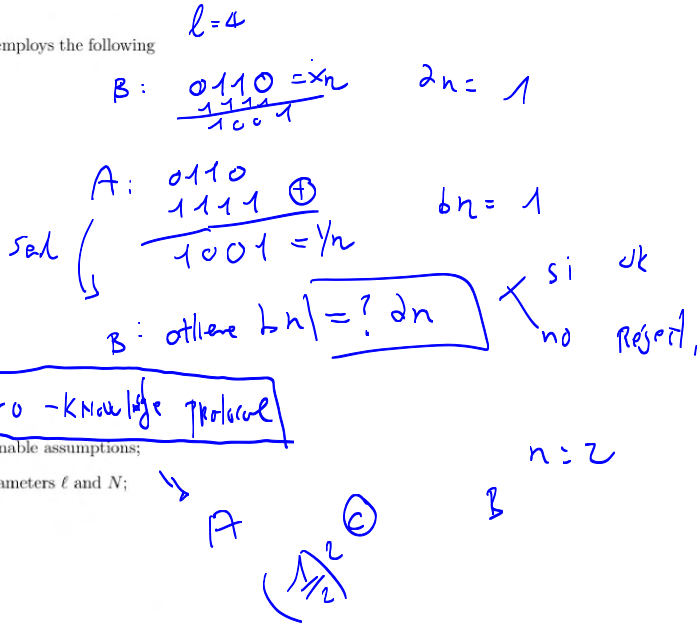
If b_n is correct for all $n = 1, \dots, N$, the authentication is accepted

3.1) show that the above protocol is correct;

3.2) identifiy its vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

3.3) evaluate the success probability of the attack in dependence of the protocol parameters ℓ and N;

3.4) suggest a possible improvement to the protocol.



Solution

Similar to the zero knowledge entity authentication process.

If E tries to guess E then the probability of success is 1/2^n which is exponentially decreasing with n so it's not a security issue.

Solution

Similar to the zero knowledge entity authentication process.

If E tries to guess E then the probability of success is 1/2^n which is exponentially decreasing with n so it's not a security issue.

1. If E can listen one step he can learn how many 1s and 0s (parity) there are in the key.

We can rewrite b as:

$b_r = y_1 \oplus \ldots \oplus y_l$

$= x_1 \oplus k_1 \oplus \ldots \oplus x_l \oplus k_l$

$= (x_1 \oplus x_2 \oplus \ldots \ldots \oplus x_n) \oplus (k_1 \oplus k_2 \oplus \ldots \oplus k_l).$

So if E can observe just one round of the protocol he can understand the parity of the key. After that he can easily authenticate since So the attack always works just by knowing a whole round of the protocol. But even without listening, an attacker can randomly assume the parity of the key and always answer to the challenge in the corresponding way, so the success in this second case is 0.5.

Jan 2020

A and B want to establish a secure connection based on symmetric cryptography, sharing a secret key k. In order to do that, since each of them has a connection with asymmetric cryptography to C, they want to use the following protocol:

k_A	private key of A
k'_A	public key of A (known to C)
k_B	private key of B
k'_B	public key of B (known to C)
k_C	private key of C
k'_C	public key of C (known to A and B)

1 A : generates nonce $r_A \sim \mathcal{U}(\mathcal{R})$
A → C : $[id_A, id_B, r_A]$

2 C : generates $k \sim \mathcal{U}(\mathcal{K})$, $u_1 = [k, r_A]$, encrypts $x_1 = E_{k'_A}(u_1)$
C → A : x_1
A : decrypts $u_1 = D_{k_A}(x_1)$, checks if r_A is consistent and obtains k

3 A : signs $t_2 = S_k([id_A, id_B, r_A])$
A → B : $x_2 = [id_A, id_B, r_A, t_2]$

4 B → C : $[id_A, id_B, r_A]$

5 C : encrypts $x_3 = E_{k'_B}(u_1)$
C → B : x_3
B : decrypts $u_1 = D_{k_B}(x_3)$, checks if r_A is consistent and obtains k

6 B : verifies $\hat{b} = V_k([id_A, id_B, r_A], t_2)$

2.1) identifiy the protocol vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

2.2) suggest changes and/or improvements to the protocol that can solve the above issues.

Solution

Summary:

- In the first 2 step, A sends to C the request to start a communication with B. In the message it is used a nonce. The message is sent in plaintext. C replies with a generated key and the nonce to confirm.
- At phase 3 A sends to B the signed nonce with its id and B sends this info to C (phase 4).
- At 5 C encrypts and sends to B the same message it sent at the beginning to A, which decrypts it, check the nonce and get the k.
- At 6 B verifies with the key the message sent by A at 3.

- First attack

A Reply attack is possible. The scheme is similar to the Needham–Schroeder protocol, which is vulnerable to the Dening-Sacco attack. The attack is based on the "known session key attack", so an attacker E is able to acquire the key of a past protocol session and masquerades as A. E replicates message 3. B will then acquire the key from C and verifies u1 with the compromised key. B will then start a session with E believing he is communicating with A.

To fix the protocol we need to make B an active participant in the protocol. To do so we can introduce an initial phase where both A and B contact each other, they may also share a nonce (included in 3) to authenticate with C as in Otway-Rees (or use timestamps as in kerberos). Then B won't accept a replied message 3 since does not started a communication request so the nonce inside 3 is not valid anymore.

- Second attack

A second attack might be a MITM, where we assume that E is a malicious user with a public and private key infrastructure in place to talk with C (which knows the public key of E). E have also access to the channel of both A and B and is able to change their plain text messages: the attack consists of 2 round of the protocol where E act as a man in the middle and is able to read messages sent from A and from B.

First run:

1. E change the message to idA,idE,rA
2. C send the key k1 to A.
3. A signs with the key k1 the messages but it E intercepts it and discards it.
4. E sends to C idA,idE,rA
5. C encrypts the key k1 and send it to E

now A and E share the key, but A thinks that only B has the key. At this point E can talk with A, which thinks he is talking with B. B never actually received any message because idB was always changed to idE.

Second run:

1. E acts as A, using idE,idB,rA
2. C send the key k2 to E.
3. E signs with the key the message idA,idB,rA
4. B sends to C idA,idB,rA, but E intercepts it and change idA to idB.
5. C send to B the key k2 to talk with E.
6. B verifies the message correctly

Same as before, but now B thinks he is talking with A, and not with E.

E can now also read the messages of both parties and even change them since he has 2 symmetrical encryption schemes in place with A and B. To block this attack a solution would be to sign (or encrypt) the plain text messages from A and B, since there is already an asymmetric key infrastructure in place.

Feb 2020

Consider the following (naïve) entity authentication protocol for a prover A who wants to prove his identity to a verifier B.

A chooses a password p_A with up to L decimal digits (that is $p_A \in \mathcal{P}$, with $\mathcal{P} = \{0,1,\dots,9\}^L$) and a one-way hash function $h : \mathcal{P} \to \mathcal{P}$. Let x_1 be the hash, and x_N be the N-fold hash, of the (possibly zero-padded) password p_A, that is

$$x_1 = h([p_A, 0, \dots, 0]) \quad , \quad x_N = \underbrace{h \circ h \circ \dots h}_{N}([p_A, 0, \dots, 0]) \, .$$

In the protocol setup phase, A securely delivers x_N to B.

Then, in the n-th run of the protocol, $n = 1, \dots, N-1$

[1] A→B : $u_1 = [\mathrm{id_A}]$

[2] B : generates challenge $c_n \sim \mathcal{U}(\mathcal{P})$
B→A : $u_2 = [n, c_n]$

[3] A : computes $x_n = h(x_{n-1})$ and $y_n = x_n + c_n \bmod 10^L$
A : builds $u_3 = [n, y_n]$
A→B : u_3

[4] B : recovers n and $\tilde{y}_n$ from u_3
B : checks that n is up to date and if so computes $\tilde{x}_n = \tilde{y}_n - c_n \bmod 10^L$
B : checks whether $\underbrace{h \circ h \circ \dots h}_{N-n}(\tilde{x}_n) = x_N$ and if so, B accepts A.

2.1) identifiy the protocol vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

2.2) suggest changes and/or improvements to the protocol that can solve the above issues.

Solution

TODO

Jan 2021

Consider the following entity authentication protocol

entities the prover A, the verifier B

tool an elliptic curve $(\mathcal{E}, \circ) \subset \mathbb{F}^2$ over the field $\mathbb{F} = \mathbb{Z}_p$ for a sufficiently large prime p, and a shared secret point $S_{AB} \in \mathcal{E}$

setup A and B take the first coordinate of S_{AB} and store it as s

[1] A : generates nonce $n_A \sim \mathcal{U}(\mathcal{R})$
A→B : $u_1 = [\mathrm{id_A}, \mathrm{id_B}, n_A]$

[2] B : randomly picks two points $P, Q \in \mathcal{E}$ as the challenge $c = (P, Q)$
B→A : $u_2 = [\mathrm{id_A}, \mathrm{id_B}, n_A, c]$

[3] A : computes points $P' = P^s$ and $Q' = S_{AB} \circ Q^s$ as the response $r = (P', Q')$
A→B : $u_3 = [\mathrm{id_A}, \mathrm{id_B}, n_A, r]$

[4] B : computes the expected response $\hat{r} = [\hat{P}', \hat{Q}']$ and, if $r = \hat{r}$, A is accepted, otherwise A is rejected

2.1) identifiy the protocol vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

2.2) suggest changes and/or improvements to the protocol that can solve the above issues.

Solution

Attack 1

If E can intercept c and set Q=P. A will then compute P' and Q' so he can listen to the reply and calculate:

(define $P'_s = (x_{P'}, -y_{P'})$)

$P'_! \circ O' = P'_! \circ S_{AB} \circ O \circ s$

$= P'_s \circ S_{AB} \circ P^{\circ s}$

$= P'_s \circ S_{AB} \circ P'$

$= S_{AB}$

So he is able to retrieve S_{AB} and break the protocol.

To avoid this attack a countermeasure can be a signature on the message with the secret, so that an attacker cannot forge the message.

Attack 2 (based on double ECC RNG attack)

given the couple P,Q if the attack knows q s.t.

$$P^{\circ q} = Q$$

then:

$Q' = S_{AB} \circ Q^{\circ s} = S_{AB} \circ P^{\circ sq}$

and since

$P' = P^{\circ s}$ we can compute

$P'^{\circ q}$, take the simmetric P'_s

and compute

$Q' \circ P'_s = S_{AB} \circ P^{\circ sq} - P^{\circ sq} = S_{AM} \circ 0 = S_{AM}$

so the secret is know to the attacker and the authentication is broken.

This attack can be avoided by selecting P and Q such that q is not known (at least to NSA :D).

Feb 2021

Consider the following protocol based on asymmetric cryptography by which two parties A and B aim at learning each other's public key

entities two legitimate parties A, and B, a trusted third party C

tools an Elgamal digital signature mechanism $(S_k(\cdot), V'_{k'}(\cdot))$ and an Elgamal encryption mechanism $(E'_{k'}(\cdot), D_k(\cdot))$

setup assume that private keys are only known to their corresponding entities, while public keys are known as follows:

- k'_{C} is known to both A and B;
- k'_{A} is known to C, but not to B;
- k'_{B} is known to C, but not to A.

1 A : generates nonce $r_{\mathrm{A}} \sim \mathcal{U}(\mathcal{R})$
A → C : $u_1 = (\mathrm{id_A}, \mathrm{id_B}, r_{\mathrm{A}})$

2 C : builds message $u_2 = (\mathrm{id_B}, k''_{\mathrm{B}})$
signs it $x_2 = S_{k_{\mathrm{C}}}(u_2)$, using r_{A} as the random exponent in the first component of the Elgamal signature
C → A : x_2

3 A : retrieves and verifies $(u_2, \hat{b}_2) = V'_{k'_{\mathrm{C}}}(x_2)$
builds message $u_3 = (r_{\mathrm{A}}, \mathrm{id_A})$ encrypts $x_3 = E'_{k'_{\mathrm{B}}}(u_3)$, using r_{A} as the random exponent in the first component of the ciphertext
A → B : x_3

4 B : $u_3 = D_{k_{\mathrm{B}}}(x_3)$
generates nonce $r_{\mathrm{B}} \sim \mathcal{U}(\mathcal{R})$
B → C : $u_4 = (\mathrm{id_A}, \mathrm{id_B}, r_{\mathrm{B}})$

5 C : builds message $u_2 = (\mathrm{id_A}, k'_{\mathrm{A}})$
signs $x_5 = S_{k_{\mathrm{C}}}(u_5)$, using r_{B} as the random exponent in the first component of the Elgamal signature
C → B : x_5

6 B : retrieves and verifies $(u_5, \hat{b}_5) = V'_{k'_{\mathrm{C}}}(x_5)$

3.1) identifiy the protocol vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

3.2) suggest changes and/or improvements to the protocol that can solve the above issues.

Solution

Since rA is transmitted in plain text, the signature of x2 is completely broken since k can be computed from

$kt_1 + r_A t_2 = v(mod p - 1)$

$k = (v - r_A t_2) t_1^{-1}$

So the attacker E can acts as C. If the attacker can access the channel and modified the messages he can forge and sign the message x2. Here E can put its own public key, instead of B one (which he acquires).

The attacker do the same for x5 with rB. A and B now talk with E with a private scheme and think they are talking with B and A, respectively.

June 2021- first

June 2021

Consider the following entity authentication protocol

entities the prover A, the verifier B

tools a symmetric MAC mechanism $(S_k(\cdot), V_k(\cdot))$

setup the key k_{AB} for the MAC mechanism is shared securely between A and B

1 A : generates ℓ-bit nonce $r_A \sim \mathcal{U}(\{0,1\}^\ell)$
A → B : $u_1 = (\mathrm{id}_A, \mathrm{id}_B, r_A)$

2 B : looks up the current time and writes it in the ℓ-bit timestamp t_B
builds message $u_2 = (\mathrm{id}_A, \mathrm{id}_B, r_A, t_B)$
signs it $x_2 = S_{k_{AB}}(u_2)$
B → A : x_2

3 A : retrieves and verifies $(u_2, \hat{b}_2) = V_{k_{AB}}(x_2)$
computes $z_{AB} = r_A \oplus t_B$ where $\oplus$ represent the XOR operation
builds message $u_3 = (\mathrm{id}_A, \mathrm{id}_B, z_{AB})$
signs it $x_3 = S_{k_{AB}}(u_3)$
A → B : x_3

4 B : retrieves and verifies $(u_3, \hat{b}_3) = V_{k_{AB}}(x_3)$
if $\hat{b}_3 = \mathrm{ok}$ and $\mathrm{id}_A, \mathrm{id}_B, z_{AB}$ in u_3 are correct, A is accepted as authentic, otherwise rejected

3.1) identify the protocol vulnerabilities and devise an attack that exploits them, under reasonable assumptions;

3.2) suggest limited changes and/or improvements to the protocol that can solve the above issues.

Solution

An adversary E can verify himself without knowing k_{AB}. Since A chooses the nounce r_A he can choose

$r_A = z_{AB} \oplus t_B$

(or $t_B - \tau$ to sync with the time E assume t_B will be generated)

Hash Functions

Mockup 2020 and 2021

Problem 1

Let $\mathcal{Y} = \mathbb{B}^\ell$ and let $f : \mathcal{Y} \mapsto \mathcal{Y}$ be a *one-way* and *one-to-one* function.
Let $\mathcal{X} = \mathbb{B}^{2\ell}$ and consider the function $h : \mathcal{X} \mapsto \mathcal{Y}$ defined as

$$h(\boldsymbol{x}) = h(x_1, \ldots, x_{2\ell}) = f([x_1, \ldots, x_\ell] \oplus [x_{\ell+1}, \ldots, x_{2\ell}]) = f(x_1 \oplus x_{\ell+1}, x_2 \oplus x_{\ell+2}, \ldots, x_\ell \oplus x_{2\ell})$$

Is h a valuable cryptographic hash function? Justify you answer.

Solution

If we recall the hash function properties:

1. easy to compute
2. given pmd of x, y should be uniformly distributed in Y
3. difficult to reverse
4. difficult given u,h(u) to find a u' s.t. h(u')=h(u)
5. difficult to find u, u' s.t. h(u)=h(u)

We can easily note that the property 4 (so 5) is not valid. Since given a message we can easily find another message with the same hash, just by swapping the first and the last parts of the message.

Feb 2020

Problem 1

Let $\mathbb{B}^\ell$, $\mathbb{B}^{2\ell}$ and $\mathbb{B}^{4\ell}$ denote the sets of binary strings with length ℓ, 2ℓ and 4ℓ bits, respectively.
Let $h : \mathbb{B}^{2\ell} \mapsto \mathbb{B}^\ell$ be a cryptographic hash function with 2ℓ-bit inputs and ℓ-bit outputs.
Define a new hash function $g : \mathbb{B}^{4\ell} \mapsto \mathbb{B}^\ell$ with 4ℓ-bit inputs and ℓ-bit outputs, for an arbitrary input $\boldsymbol{x} = [x_1, \ldots, x_{4\ell}]$ by repeating h as follows

$$\begin{aligned}[y_1, \ldots, y_\ell] &= h([x_1, \ldots, x_{2\ell}]) \\ [y_{\ell+1}, \ldots, y_{2\ell}] &= h([x_{2\ell+1}, \ldots, x_{4\ell}]) \\ g(\boldsymbol{x}) = [z_1, \ldots, z_\ell] &= h([y_1, \ldots, y_\ell, y_{\ell+1}, \ldots, y_{2\ell}])\end{aligned}$$

or equivalently

$$g(\boldsymbol{x}) = h([\,h([x_1, \ldots, x_{2\ell}]),\ h([x_{2\ell+1}, \ldots, x_{4\ell}])\,])$$

1.1) Prove that if $h(\cdot)$ is pre-image resistant, so is $g(\cdot)$

1.2) Prove that if $h(\cdot)$ is strongly collision resistant, so is $g(\cdot)$

Solution

The idea is similar to the Merkle Damgard construction.

- 1.1 we prove that if g is not pre-image resistant then h is not preimage resistance.

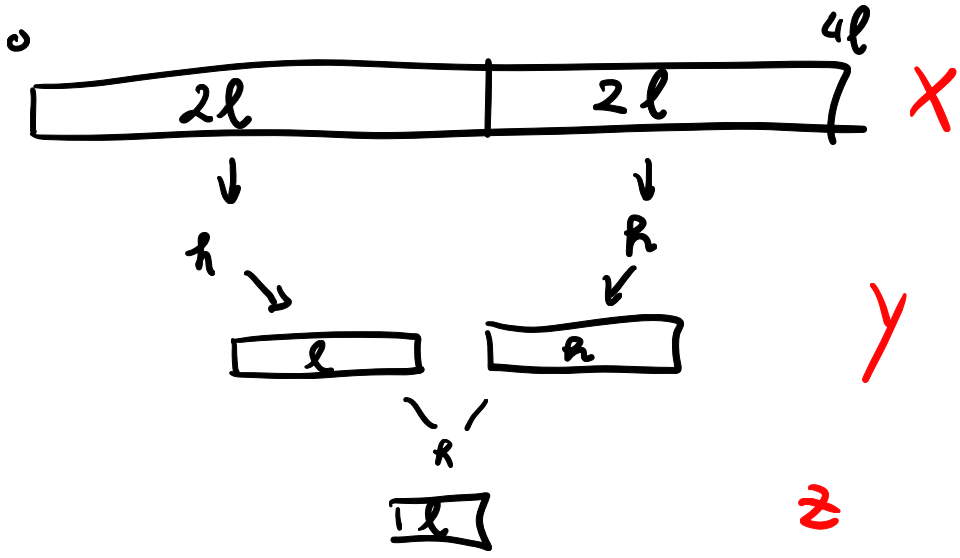
If g is not preimage resistant then we can find $x = g^{-1}(v)$. We can now divide x in $[x_1, \ldots, x_{2l}]$ and $[x_{2l+1} \ldots x_{4l}]$ get their hash $[y1, \ldots, y_l]$ and $[y_{l+1}, \ldots, y_{2l}]$. But the hash of the concatenation of this 2 hash is v, so we were able to reverse h in the final step of g, so h is not preimage reistance.

$h: B^{2\ell} \to B^{\ell}$ $\quad h: Y \to Z$

$g: B^{4\ell} \to B^{\ell}$ $\quad g: X \to Z$

$g(x_1, x_2 \dots x_{4\ell}) = h(y_1 \dots y_{2\ell})$

dove $[y_1 \dots y_\ell)$ sono output $h(x_1, x_2 \dots x_{2\ell})$.



Prove: se h è good hash function (Resistenza, collision resistance.)

$\downarrow$

g è good hash function.

Per contrazione: (Resistenza)

se h fosse non sicura, allora è facile trovare una contro immagine di z in Y.

Di conseguenza anche g non è sicura in quanto usa la funzione h. (è facile trovare una contro immagine)

- Same for collision.

- 1.2 similar, is g is not collision resistant so it's not h.

If g is not collision resistant we can find a pair x,x' s.t. g(x)=g(x'). We compute the first step of g for both and we get y and y', whose hash are g(x) and g(x'), so, since g(x)=h(y) and g(x')=h(y') then we found a colliding pair y,y' for h, which is not collision resistant.

Feb 2021

Problem 1

Given a binary string $a \in \{0,1\}^n$ let $\bar{a}$ denote its *bitwise complementary*, that is

$$\bar{a} = a \oplus [1,1,\ldots,1]$$

Consider a cryptographic hash function $h : \{0,1\}^\ell \mapsto \{0,1\}^n$, and by modeling it as an ideal random function,

1.1) prove that the probability of finding at least one pair of complementary hash outputs among L distinct input strings is the same as that of finding at least one pair of colliding hash outputs

1.2) compute upper and lower bounds to the above probability for $\ell = 1024$, $n = 100$, $L = 10^{15}$

1.3) assuming a single hash can be computed in $T_h = 10\,\mu$s and neglecting all other computation (e.g., sorting, comparing) times, state whether the hashing function h with $\ell = 1024$, $n = 100$, is (ε, T_0) computationally secure against collision attacks, with $\varepsilon = 10^{-6}$ and $T_0 = 1$ year

XOR { 1 0 → 1 ; 0 1 → 1 ; 1 1 → 0 ; 0 0 → 0

Solution

- 1.1

If the space of hash function is $V = \{0,1\}^n$ and all the hash are equally probable, then if $v = h(u)$ and $v' = h(u')$ for n distinct pair:

$P(v = v') = P(v = v' \oplus [1,1,\ldots 1])$ which is equal to $1 - P(v_1! = v_2 \oplus [1,1,\ldots 1]! = v_3 \oplus [1,1,\ldots 1]! = \ldots = v_n \oplus [1,1,\ldots 1])$

We can evaluate $P(v = v' \oplus [1,1,\ldots 1]) = P(v = v' \oplus \bar{1})$ at least for a pair as

$$1 - P(v_1! = v_2 \oplus \bar{1})P(v_1! = v_3 \oplus \bar{1}) \ldots (P(v_2! = v_3 \oplus \bar{1}) \ldots$$

$$1 - \sum_{a \in V} P(v_1 = a \oplus \bar{1}) \sum_{a \in V - v_1} P(v_2 = a \oplus \bar{1}) \ldots \sum_{a \in V - v_1, \ldots, v_{n-1}} P(v_n = a \oplus \bar{1})$$

Since the hash are equally probable the probability to find a hash in V is $P_v(a \oplus \bar{1}) = \frac{1}{|V|}$ and the sums have respectively $|V|, |V|-1, \ldots, |V|-n+1$ terms, then:

$$1 - \prod_0^{n-1}(1 - \frac{i}{|V|})$$

Which is the same probability to find a collision among n hash digests.

- 1.2

This probability can be upperbounded since:

$$\prod_0^{n-1} \frac{i}{|V|} = \frac{n(n-1)}{|V|} = \frac{n^2 - n}{2|V|}$$

$$1 - e^{-\frac{n^2-n}{2|V|}} \leq 1 - \prod_0^{n-1}(1 - \frac{i}{|V|}) \leq \frac{n^2 - n}{2|V|}$$

if $n^2 << |V|$

the lowerbound has value $1 - e^{-\frac{10^{15*2} - 10^{15}}{2*2^{100}}} = 0.326$

and the upperbound $\frac{10^{15*2} - 10^{15}}{2 \times 2^{100}} = 0.394$

- 1.3

In one year we can compute $N_h = 365 * 24 * 60 * 60/(10 * 10^{-6}) = 3.15 * 10^{12}$ hash. According to the upper bound probability, the probability of success with Nh pairs is between 3.913*10^-6 and 3.914*10-6, so the the alghorithm is not epsilon computationally secure.

Random Number Generators

Mockup 2020 and 2021

Problem 2

SL = SECURITY LEVEL.

Consider a mechanism M for cryptographic key generation that, every time it is called, outputs a random and independent string k of ℓ binary digits, uniformly drawn among all the strings with the same number $\ell/2$ of 0s and 1s.

2.1) Explain which of the following statements is true about the security level of M against attacks that aim to guess the value of the generated key k:

a) $\mathrm{SL}(M) \geq \ell$ bits;

b) $\mathrm{SL}(M) < \ell$ bits;

2.2) Compute the exact value of $\mathrm{SL}(M)$ in the case $\ell = 16$

$P(S_A | A, M) = 5\%$

- Some unnecessary math:

The space of k with l/2 0s and 1s is the $\binom{l}{l/2} = \frac{l!}{l/2!l/2!} = \frac{(l)(l-1)..(l/2+1)}{l/2!}$

and the probability to guess a key in this set is:

$P_s = \frac{l/2!l/2!}{l!} = \frac{l/2!}{(l)(l-1)..(l/2+1)} \rightarrow \frac{1}{\binom{l}{l/2}}$

so the security level is:

$SL(M) = -log_2 \frac{l/2!l/2!}{l!} = -(log_2(l/2!) + log_2(l/2!) - log_2(l!))$

Solution

if $SL(M) = l$ then:

$log_{1/2} max P_s = l$

$max P_s = 1/2^l$.

However since all the bits are not randomly selected, but the numbers of 1s and 0s are fixed, then the probability of success can't be $1/2^l$ which would be the probability of success for a random guess in the key of randomly chosen 0 and 1 with no constraints. The option b) is the correct one.

if $l = 16$

$SL(M) = -log_2(\frac{16!}{8!8!}) = \frac{8*7*6*5*4*3*2*1}{16*15*14*13*12*11*10*9} = log_{1/2}(1/12870) \sim 13.65$

Feb 2021

Problem 2

A true random number generator outputs binary independent and non uniform symbols $\{z_n\}$ with distribution $p_z(0) = 51\%$, $p_z(1) = 49\%$

2.1) Prove that a sequence of $N = 10$ symbols from the generator is ε-unconditionally secure for $\varepsilon = 0.04$

2.2) Assume the sequence is used as a one-time-pad encryption key for a 10-bit message, where 5 uniformly distributed bits are repeated twice, and compute an upper bound to the success probability of a guessing attack on the message

Solution

- 2.1 An attacker tries to guess the key. Since the 0 are more probable he will try to guess the key to be $\{0\}^{10}$, which is the most probable key.

However the probability of that key is $(0.51)^{10} = 1.19 \times 10^{-3}$ so the generator is ϵ-unconditionally secure.

- 2.2

If the message is constructed as $[x_1, .., x_5] = [x_6, \dots, x_{10}]$, with $x_i = x_{i+5}$.

If we focus, for example, on x_1 and x_6, we can note that

$x_1 \oplus x_6 = u_1 \oplus k_1 \oplus u_6 \oplus k_6 = k_1 \oplus k_6$.

- Idea for a smart attack:

$b_1 = x_1 \oplus x_6 = k_1 \oplus k_6$

if $b_1 = 0$ then we know that the 2 key bits are equal. Since it's more probable that it's two 0s, the attacker will try to forge a key to retrieve the message choosing $k_1 = 0$. This key will be composed by a 0 where $b_i = 0$ and will be randomly chosen between a 0 and a 1 where $b_i = 1$.

> NOTE: if $b_i = 1$ it's not more convenient to choose the key bit equal 0, because b_i is the result of a xor between a 0 and a 1 or between a 1 and a 0, equally probable with probability $0.51 * 0.49$.

First we consider

$P(x_1 = x_6) = P(k_1 = k_6) = P(k_1 = 0, k_2 = 0) + P(k_1 = 1, k_2 = 1) = 0.51^2 + 0.49^2 = 0.5002$

and

$P(x_1 != x_6) = P(k_1 != k_6) = P(k_1 = 1, k_2 = 0) + P(k_1 = 0, k_2 = 1) = 2 * 0.49 * 0.51 = 0.4998$

if $x_1 = x_6$ then $k_1 = k_6$, and they are two 0s with probability 0.51^2 and two 1s with probability 0.49^2. Instead if $x_1 != x_6$ then $k_1 != k_6$ and the probability the of "01" is equal to "01" so the attacker guess 0 or a 1 and guess with 0.5 probability.

Since if $x_1 = x_6$ it is more probable that $x_1 = x_2 = 0$, so the attacker guess they are two 0.

For a message with p couples of equal bits, the attack has a guessing probability of:

$P_{s|p}(p) = (0.51 * 0.51)^p (0.5)^{5-p} = (0.2601)^p (0.5)^{5-p}$

The probability of success is (total probability theorem):

$P_s = \sum_{a=0}^{5} P_{s|p}(p = a) * P(p = a) = \sum_{a=0}^{5} P_{s|p}(p = a) * P(x_1 = x_6)^a P(x_1 != x_6)^{5-a} = 1.99 * 10^{-3}$

So an upper bound to the attack is $2 * 10^{-3}$

AWGN

Jan 2020

Consider a discrete time wiretap AWGN channel in which

- the symbol time is $T = 1\,\mu s$
- the legitimate channel has an amplitude gain $g = 1/100$
- the eavesdropper channel has an amplitude gain $h = 1/500$
- the noise has variance $\sigma^2 = 10^{-6}\,V^2$ for both the legitimate receiver and the eavesdropper

3.1) Is it possible to build an encoder/decoder pair that provide physical layer secrecy for a message with rate $R_u = 2\,Mbit/s$? Justify your answer.

3.2) If your answer to the above question is "YES", find the minimum statistical power of the transmitted symbols that is necessary; if your answer to the above question is "NO", find the minimum gain of the legitimate channel that is necessary so that physical layer secrecy at rate R_u is possible

Solution

- 3.1 With a rate R_u and T=1μs we want a secrecy rate $R_s = T * R_u = 2bit$.

We can compute the channel secrecy as

$C_s = \frac{1}{2} log_2 \frac{\bar{\Lambda_y}}{\bar{\Lambda_z}}$

$= \frac{1}{2} log_2 \frac{g_y^2/\sigma^2}{g_z^2/\sigma^2} = \frac{1}{2} log_2 25 = log_2 5 = 2.32$

So since C_s is the maximum achievable secrecy rate, we can get a $R_s = 2bit$ secrecy rate.

- 3.2 we can find the minimum statistical power P as:

$R_s = 2 = \frac{1}{2} log_2 \frac{1+\Lambda_y P}{1+\Lambda_z P}$

$\frac{1+\Lambda_y P}{1+\Lambda_z P} = 16$

$1 + \Lambda_y P = 16 + 16(\Lambda_z P)$

$1 + \Lambda_y P = 16 + 16(\Lambda_z P)$

$(\Lambda_y - 16\Lambda_z)P = 15$

$P = \frac{15}{(\Lambda_y - 16\Lambda_y)} = \frac{15}{(100 - 16 \cdot 100/25)} = 540V^2$

Jan 2021

Problem 3

Consider a discrete time wiretap BSC channel A → B, E in which

- the symbol time is $T = 0.5\,\mu s$
- the legitimate channel A → B has a symbol error rate $\varepsilon = 1\%$
- the eavesdropper channel A → E has a symbol error rate $\delta = 2\%$

3.1) Is it possible to build a key agreement scheme that exploits a two-way public error-free channel A ↔ B to provide A and B with a new pair of unconditionally secure symmetric keys of length $\ell_k = 256$ bit, every $T_k = 2\,ms$? Justify your answer.

3.2) Independently of your previous answer, in a key agreement scheme for this channel would you rather have *information reconciliation* performed by modifying the information at B to match the one at A, or viceversa? Justify your answer.

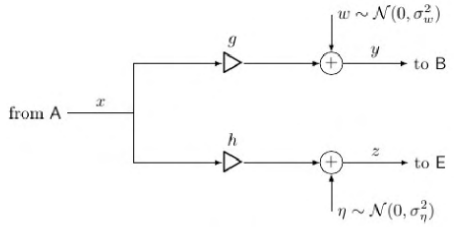
Solution

$C = h(\gamma) - h(\epsilon) = h(\epsilon + \gamma - 2\epsilon\gamma) - h(\epsilon)$

TODO

June 2021

Problem 2



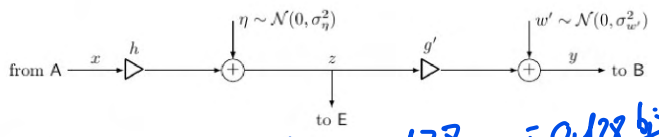
Consider a discrete time wiretap AWGN channel as pictured above, in which

- the symbol time is $T = 1\,\mu s$
- the legitimate channel has an amplitude gain $g = 10^{-3}$ and noise variance $\sigma_w^2 = 2 \cdot 10^{-5}\,V^2$
- the eavesdropper channel has an amplitude gain $h = 10^{-2}$ and noise variance $\sigma_\eta^2 = 10^{-3}\,V^2$

2.1) Is it possible to build a key agreement scheme that exploits a two-way public error-free channel A ↔ B to provide A and B with a new pair of unconditionally secure symmetric keys of length $\ell_k = 128$ bit, every $T_k = 1\,ms$? Justify your answer.

2.2) Repeat the above question under a maximum power constraint $P_{max} = 1\,V^2$ for the signal transmitted over the AWGN channel.

2.3) Repeat question 2.1 for the wiretap channel depicted below, where $g' = 10^{-1}$, and $\sigma_{w'}^2 = 10^{-5}\,V^2$, while h and σ_η^2 are the same as above.



Solution

TODO

Message Auth and Integrity protection

Feb 2020

Problem 3

A symmetric mechanism for message authentication and integrity protection employs 128-bit keys and 32-bit tags. Compute a lower bound to the maximum success probability of:

3.1) an attack aiming to forge one single message

3.2) an attack aiming to forge 3 consecutive messages

3.3) an attack aiming to forge 10 consecutive messages

Solution

- 3.1

A brute force attack on a single message will succeed with a probability $P_s(A_m^1) = \frac{1}{2^{32}}$.

- 3.2

A brute force attack on 3 consecutive messages will succeed with a probability $P_s(A_m^3) = (\frac{1}{2^{32}})^3 = \frac{1}{2^{96}}$.

- 3.3

Since a brute force attack on 10 consecutive messages will succeed with a probability $P_s(A_m^{10}) = (\frac{1}{2^{32*10}})$, the attacker will try to guess the key instead, with a probability to succeed of $P_s(A_k) = \frac{1}{2^{128}}$.

Jan 2021

Problem 1

Consider a security solution for message authentication and integrity protection that is built as follows:

a) the signing and verification mechanism uses strongly universal$_2$ hashing + one time pad on the authentication tag

b) the keys for the signing mechanism in a) are provided by an Elliptic Curve Diffie-Hellman key agreement protocol

c) the random secrets for the key agreement protocol in b) are provided by quantum random number generators (QRNG)

and answer the following questions, with adequate justification

1.1) does the overall system provide unconditional security? computational security? post quantum security?

1.2) how can it be protected against man-in-the-middle attacks on the key agreement protocol?

Suppose that information messages are $\ell_u = 5000$ bit long and are transmitted one every $T_u = 1$ ms, and aiming at making the probability of successfully forging each single message by a computationally limited adversary no larger than $P_{\max} = 10^{-12}$

1.3) find the minimum length (in bit) of each authentication tag

1.4) find the minimum output rate (in bit/s) of the QRNGs that is necessary

Solution

- 1.1 since ECDH can be solved in polynomial time by a quantum computer, the system does not provide post quantum security. Unconditional security is provided by the one time, so computational security.
- 1.2 to avoid man in the middle the two parties can use a digital signature scheme to protect the messages cA and cB. The scheme can be Elliptic curve DSA, since an elliptic curve scheme is already in place.
- 1.3 since we need that (univeral2 hash function)

$H(t) = log_{1/2}\epsilon$

and

$H(t) = E[i(t)] = log_{1/2} 1/|T| \geq log_{1/2}\epsilon$

$1/|T| = \epsilon$

$|T| = 2^t \geq 10^{12}$ and $t = ceil[log_2 10^{12}] = 40bit$

only if $|M| = 2^{5000} >> |T| = 2^{40}$

- 1.4 $rate = 40/1ms = 4 \times 10^4$ since the one time pad needs a different key for every tag.

June 2021

Problem 1

A binary block cipher with block size ℓ for both plaintext and ciphertext is used in a CBC-MAC scheme to authenticate and protect the integrity of L-bit messages (with $L \gg \ell$).

1.1) Prove that if the block cipher encryption function $E_k(\cdot)$ is modeled as an ideal random permutation, then the CBC-MAC tag computation function $T_k(\cdot)$ is equivalent to an ideal random function.

1.2) Explain why the CBC-MAC tag computation function is not equivalent to a random permutation itself.

1.3) Compute the probability that two arbitrarily chosen L-bit messages have the same MAC, in the ideal model from question 1.1 with $\ell = 128$ and $L = 1024$.

Solution

TODO

Misc

Jan 2020

Given a binary vector $k \in \mathbb{B}^\ell$, let $C_k \in \mathbb{B}^{\ell \times \ell}$ denote the square circulant matrix having k as its first row, with each row being the cyclic shift of the previous row by one step to the right. For instance if it were $\ell = 6$ and $k = [101100]$, it would be

$$C_k = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Assume A and B have shared in advance a long stream of random, independent and uniform binary digits and they devise the following mechanism for secret communication.

- plaintext, ciphertext and key are ℓ-digit binary strings $\mathcal{M} = \mathcal{X} = \mathcal{K} = \mathbb{B}^\ell$, the plaintext is uniform in $\mathcal{M}$
- for the encryption of each message u, a new key k is taken by reading ℓ consecutive bits from the secret stream; if the binary matrix C_k obtained from k is invertible, then A and B use it for encrypting u (see below); otherwise they discard the current k and try again with the next ℓ bits in the stream.
- the ciphertext is computed as $x = C_k \odot u$ where $\odot$ denotes matrix product in the binary field

1.1) Choose the decryption function $D(k, x)$ so that the system offers *perfect recovery* of the secret message u.

1.2) Does the system offer *perfect secrecy* for any value of the parameter ℓ? Prove your statement.

Solution

TODO