Randomness and Cryptographic Security

Teórica #2 de Criptografia Aplicada

Chaves Criptográficas (Cryptographic Keys)

De onde aparecem estas chaves?

Em **criptografia simétrica**, podem ser:

- Geradas de forma uniforme usando um método aleatório ou com base numa outra chave conhecida (pre-shared key);
- Derivadas de uma função de derivação de chaves (KDF - Key Derivation Function), com base em:
  - Uma password ou chave secreta de **baixa entropia**;
  - Uma chave mestre de **alta entropia** com base num protocolo de partilha de chaves (KEP - Key Exchange Protocol)

Em **criptografia assimétrica**, podem ser:

- Geradas através de um algoritmo de chave pública-privada, onde este gera duas chaves que se complementam entre sim, sendo geradas aleatoriamente e uniformemente. A chave pública é conhecida mas a privada só conhecida por quem a detém. Por norma estas chaves são muito maiores: Chaves RSA tem cerca de 4096 bits para o modo 128-bit security e chaves Elipitic-Curve tem cerca de 400 bits também para o modo 128-bit security.

Como são estas chaves geradas e guardadas ?

Idealmente num ambiente seguro de hardware, como por exemplo:

- Hardware Security Module (HSM): Componente hardware responsável por encriptar e decriptar de forma segura chaves criptográfica, também como armazenar estas;
- Smartcard ou *token* criptográfico.

Chaves que permanecem armazenadas durante periodos temporais grandes, por norma são ***wrapped*** com outra chave, antes de serem armazenadas, usando Password-Based Encryption (Low Security), Hardware-Protected Master Key (Standard Security) ou uma chave mestre que pertence ao dispositivo hardware seguro (High Security).

Aleatoriedade (Cryptographic Keys)

O que é aleatoriedade (randomness) ?

Aleatoriedade não é uma propriedade de uma string (no sentido em que não é possível dizer que uma string é mais aleatória que a outra). É sim uma propriedade de:

- Um processo de geração (randomised processes);
- Um procedimento de *sampling* (sampling procedure).

Processos de geração são descritos através de **distribuições aleatórias** (randomness distributions). Estes começam através uma distribuição uniforme, descrita ao longo de um conjunto finito `S`. Um processo `U` recolhe uma amostragem (samples) da distribuição uniforme se:

$$\forall s^* \in S, \Pr[s = s^* : s \twoheadleftarrow U] = \frac{1}{|S|}$$

Strings com tamanho `λ`:

- Qualquer string ocorre com uma probabilidade `1/2^λ`
- Podem ser implementadas ao descrever cada bit como uma "moeda perfeita".

O que é Entropia (Entropy) ?

Entropia mede a **incerteza** de uma amostra.

O que são Random Number Generators (RNG) ?

Processos combinados que combinam:

- Uma fonte de entropia (definida como bit string);
- Extratores de aleatoriedade (randomness extractores) (por norma uma função hash), que comprimem as bit strings de entropia em `λ` bits.

Estes processos são tipicamente lentos, dado que boa aleatoriedade é difícil de se conseguir. Dado isto, algoritmos RNG são não determinísticos.

O que são Pseudorandom Number Generators (PRG) ?

PRG são a resposta ao problema de os RNG serem lentos. Estes apenas consideram uma pequena amostra aleatória e uniforme de tamanho `λ` (seed) e geram bit strings longas aleatórias. Estes algoritmos são por si determinísticos.

Um atacante não deve conseguir distinguir PRGs de uma string aleatória, sem saber o seu input (seed).

Stateful RNG in Operating Systems

Em sistemas operativas modernos, geração aleatória contém estado:

- O algoritmo PRG mantem o estado, e o sistema operativo mistura a entropia com o estado PRG. O fluxo do algoritmo pode ser representado da seguinte maneira:

init() -> PRG -> refresh(entropy) -> refresh (entropy) -> ... -> next() (termina o algoritmo e devolve a bit string aleatória.

Sistema operativas devem estar cientes do compromisso do estado do PRG. Este deve ser resistente a backtracking e prediction.

Em sistemas `*nix`, PRG é acessível através do ficheiro `/dev/urandom`. Em alguns sistemas `*nix`, existe também o ficheiro `/dev/random` que permite obter as a aleatoriedade com a contrapartida que o acesso a este bloqueia no caso de não haver mais entropia.

Quantifying Cryptographic Security

Segurança criptográfica é quantificada matemática e esta tenta *definir o impossível*. O impossível é definido como nenhum atacante tem vantagem sobre o algoritmo, independentemente do seu poder computacional, diante um certo valor (epilson).

O que é o One-Time-Pad (OTP) ?

Esquema de encriptação que usado corretamente é impossível de ser quebrado. Utiliza como chave uma permutação aleatória sobre o texto a cifrar, no qual esta chave só pode ser usada uma vez, caso contrário o algoritmo pode ser quebrado. Esta permutação é igual à da cifra de Vinenère, sendo que a chave tem o mesmo tamanho que o texto. Utiliza a operação bit-wise XOR para encriptar e decriptar o texto.

Vantagens: Inquebrável, se não conhecida a chave Desvantagem: Requer chaves tão grandes como o texto a cifrar e o armazenamento destas torna-se difícil. As chaves apenas só podem ser usadas uma vez.

Contudo, as ideia atrás do OTP continua a ser central à criptografia moderna:

- Prova que uma bit string é “tão boa” como uma string aleatória;
- Permite a utilização da própria bit string como chave OTP.

Esquemas de Encriptação Atuais

Hoje em dia um boa esquema de encriptação é designado por boas permutação de chave, que:

- São chamadas de block cipher ou permutações pseudo aleatórias;
- Requerem como input uma grande quantidade de bits (e.g., 256 bits)
- Requerem como chave uma grande quantidade de bits (e.g., 256 bits)

A encriptação em si é realizada através de uma aplicação gradual de um processo iterativo, que utiliza a permutação desenhada e a chave repetidamente. Este processo tem de nome *um modo de operação* (mode of operation).

Cifra de Bloco Ideal

A cifra de bloco ideal é uma sampled permutation uniformemente e aleatória.

Desvantagens? Requer uma tabela enorme contendo as chaves, que não se consegue comprimir numa descrição compacta.

Para tal utiliza-se block ciphers, também conhecidas como permutações pseudo aleatórias, pois:

- Contem uma descrição pública compacta;
- São computáveis eficientemente;
- Requerem uma chave secreta pequena.

Security Assurance

Como assegurar segurança para esquemas n-bit?

Duas vertentes:

- **Heuristic Security**:
  - Comunidade de cryptanalysts que tentam quebrar o esquema (Prova Prática). Congressos/Competições para quebrar os algoritmos que cada um especificou.
- **Provable Security**:
  - Prova matemática (Prova teórica/formal), onde quebrar o esquema implica resolver um “hard problem”.