ENPM 809W Introduction to Secure Software Engineering

Gananand Kini

Lecture 5

Cryptography related security bugs - Attacks

Outline

- Introduction to Cryptography Core Knowledge
- Cryptography weaknesses in software systems

Outline

- **Introduction to Cryptography Core Concepts**
  - Hashing
  - Encryption
  - Symmetric vs Asymmetric Ciphers
  - Simple vs Block-based vs Stream based
  - Encryption Modes
  - Use of random numbers
- **Patterns can be bad**
- **RSA algorithm attack Brumley**
- **Android “Master Key” Use what you check**
- **Hash Password Cracking**
- **Unprotected Transport of Credentials (for example no HTTPS) and Attacker-In-The-Middle**
  - Packet Capture using Wireshark/Message capture using HTTP Proxy
  - POODLE attack
- **Seeding and Flavors of Random Numbers**

Introduction to Cryptography Core Knowledge

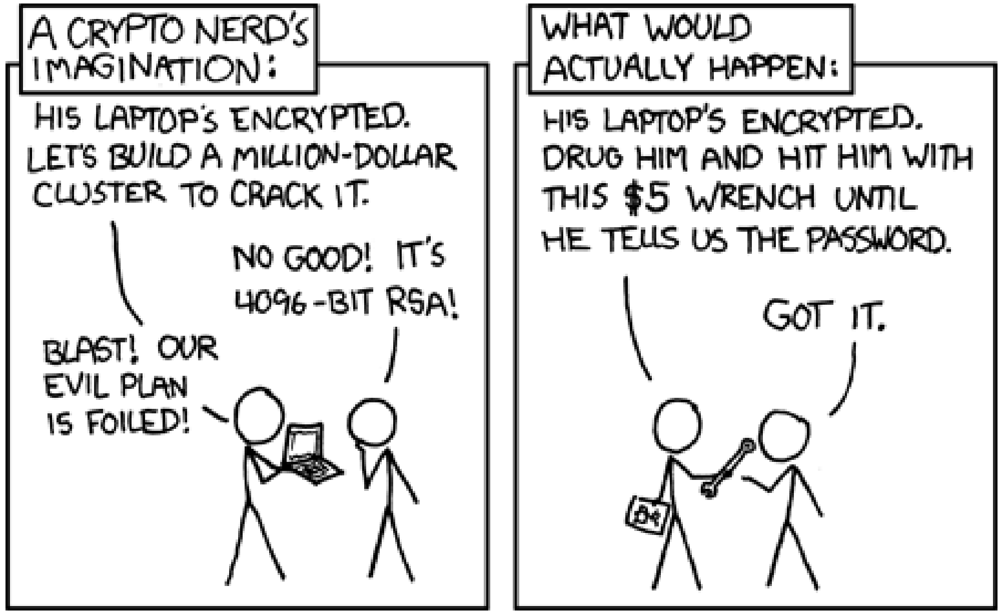
Cryptography and Cryptanalysis

- **Cryptography [1]: The discipline that embodies the principles, means, and methods for the transformation of data in order to hide their semantic content, prevent their unauthorized use, or prevent their undetected modification.**
- **Cryptanalysis [2]: The study of mathematical techniques for attempting to defeat cryptographic techniques and/or information systems security. This includes the process of looking for errors or weaknesses in the implementation of an algorithm or of the algorithm itself.**
- **These are just one definition among many.**
- **Cryptography tries to transform input symbols from one space to output symbols in another space such that the transformation might be known, but cannot be reversed, cannot be hinted or even known ahead of time.**
- **Cryptanalysis tries to do just the opposite.**
- **We are trying to implement Cryptography (defense) while trying to make Cryptanalysis (attack) very difficult.**

Sources:
1. NIST Glossary: Cryptography. https://csrc.nist.gov/glossary/term/cryptography. Retrieved July 10, 2021.
2. NIST Glossary: Cryptanalysis: https://csrc.nist.gov/glossary/term/cryptanalysis. Retrieved July 10, 2021.

What its about and what its NOT about …

- **Cryptography is a tool in a toolbox that can help:**
  - Secure data
  - Inform about properties of data
  - Prevent unauthorized use or modification of data
- **Cryptography is not:**
  - Going to solve all security problems
  - Something you just "add on" for security
- **Would rather have an application that is usable than deal with complex security that impede the software system functionality for legitimate users.**


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Sources:
1. Randall Munroe. XKCD 538: Security. xkcd: Security. Retrieved July 10, 2021.

Can you /should you “roll” your own cryptography scheme?

- **No. A grave disservice and injustice to the users of the software system.**
- **Requires a PhD and advanced training in Mathematics and Cryptography.**
- **Instead strategy is to reuse what exists based on whether it is considered <u>safe</u> for the designed use case.**
- **Tiny implementation errors can have disastrous impacts on the software system and its security!**

Sources:
1. Overlord. Wile E. Coyote A Hero, Of Sorts. https://coyotepr.uk/films/wile-e-coyote-hero-sorts/. Retrieved July 10, 2021.

Common uses in security for cryptographic algorithms

- Confidentiality/Secrecy – Keeping data confidential or secret.
- Integrity – Ensuring and sometimes proving data has not been tampered.
- Non-repudiation – Ensuring sender of information is provided with proof of delivery and recipient is provided with the proof of sender's identity, such that neither can later deny having processed that information.

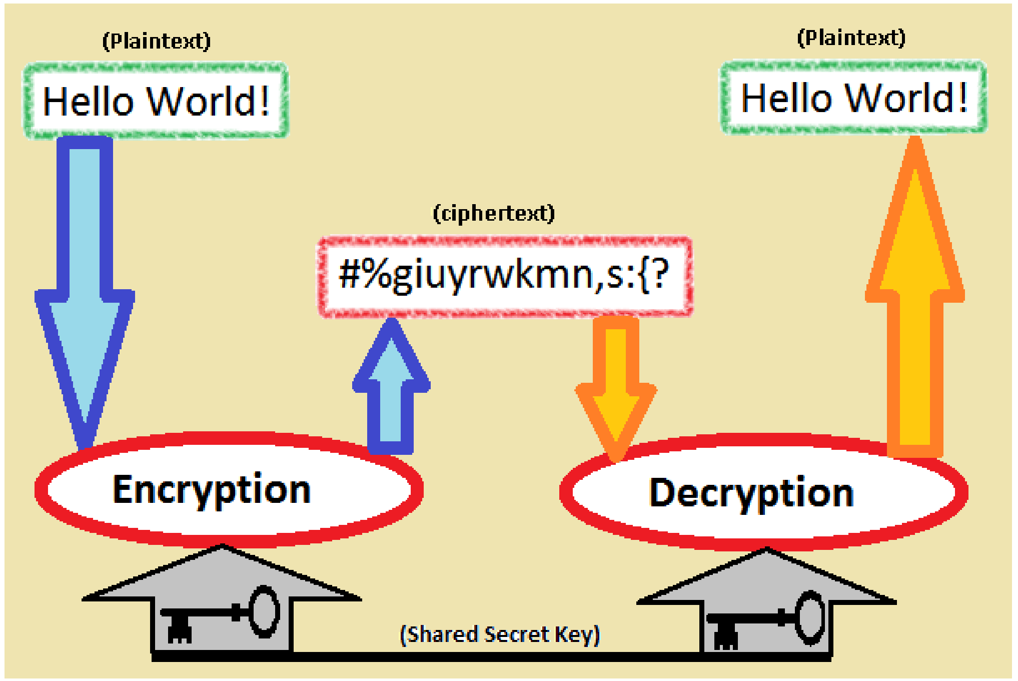
Sources:
1. NIST Glossary: Non-repudiation. https://csrc.nist.gov/glossary/term/non_repudiation. Retrieved July 10, 2021.

Common basic types of cryptographic algorithms

- **Symmetric (key) based**
  - E.g. ~~DES~~, ~~3DES~~, Advanced Encryption Standard (AES) (strikethrough ones are old and easily broken)
- **Asymmetric (key) based**
  - E.g. RSA, Elliptic Curve family of algorithms
- **Encryption can be stream based or block based**
  - Stream based encryption operates on streaming data (just like FileStream InputOutputStream etc.)
  - Block-based divides up data into blocks and then encrypts each block.
- **Hash (a.k.a. "digital fingerprint")**
  - Implemented using one-way functions like MD5, SHA-1, SHA-256 etc.
- **Cryptographic protocols – an abstract or concrete protocol that performs security-related functions by applying cryptographic methods.**
  - Diffie-Hellman (Key exchange)
  - Curve25519 (Elliptic Curve X25519 function Based Diffie-Hellman or ECDH key exchange)
  - Transport Layer Security (TLS used for Secure Sockets Layer internet protocol)
  - Etc.
- **Random Number Generation**

Symmetric Encryption/Decryption

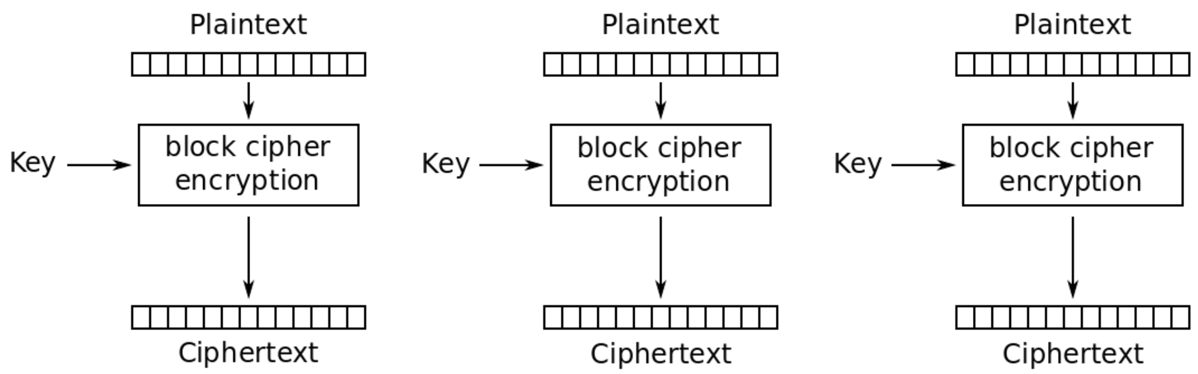
- **Symmetric key algorithms use a shared "secret" key to perform both:**
  - The encryption and,
  - Later decryption of data.
- **Symmetric Key Block based algorithms (like AES) can have modes:**
  - ECB or Electronic Code Book mode – should not be used if data is longer than one block
  - CBC or Cipher Block Chaining, CFB or Cipher Feedback, OFB or Output Feedback – if you need only encryption and not decryption to save on code space
  - CTR or Counter mode – Better parallelization
  - XTS mode - If encoding random access data like HDD or RAM
  - OCB or Offset Code Book mode – authenticated encryption by Phillip Rogaway, Mihir Bellare, John Black and Ted Krovetz.
- **If you have to pick one – pick the OCB3 (RFC 7253) mode (it provides both message authentication and privacy/confidentiality).**



Sources:
1. Dev-NJITWILL. Wikipedia: Symmetric Key Algorithm. https://simple.wikipedia.org/wiki/Symmetric-key_algorithm. Retrieved July 10, 2021.

Block based cryptographic algorithms: ECB mode

- **Electronic Code Book**
- **The simplest of the encryption modes.**
- **Takes block of plaintext and a key.**
- **Very weak since this lacks the property of diffusion.**

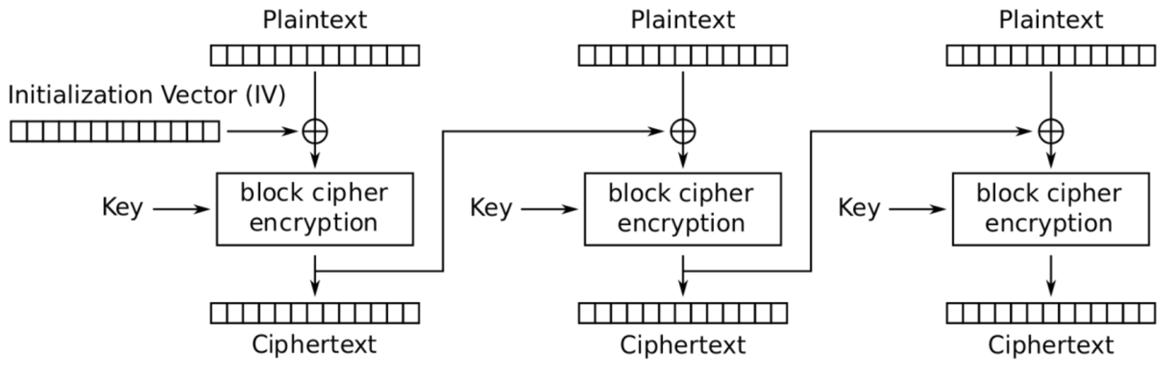


Electronic Codebook (ECB) mode encryption

Sources:
1. Wikipedia: Block Cipher mode of operation. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation. Retrieved July 10, 2021.

Block based cryptographic algorithms: CBC Mode

- Cipher Block Chaining
- Take the output cipher text from one block and apply it to encryption of next block.
- First block uses an Initialization Vector to start the process.
- The process is chained in this manner until the last block.

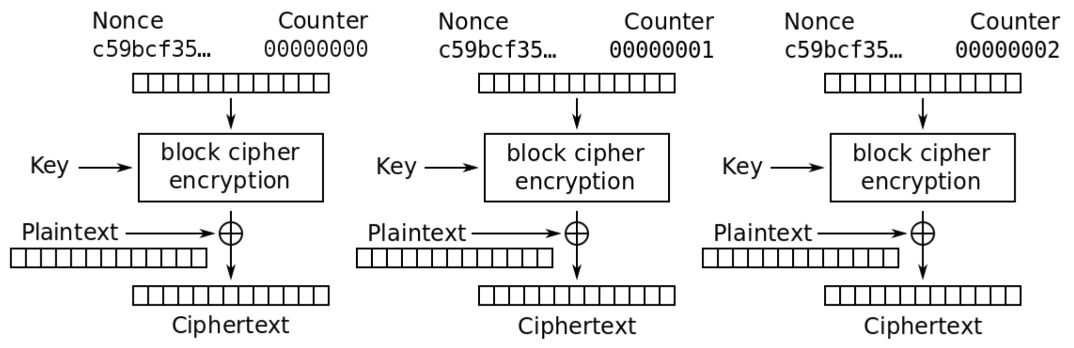


Cipher Block Chaining (CBC) mode encryption

Sources:
1. Wikipedia: Block Cipher mode of operation. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation. Retrieved July 10, 2021.

Block based cryptographic algorithms: CTR Mode

- Counter mode
- Allows for random access property during decryption because of the counter.
- Can be parallelized across multiple cores.
- Nonce is combined with the counter using concatenation.



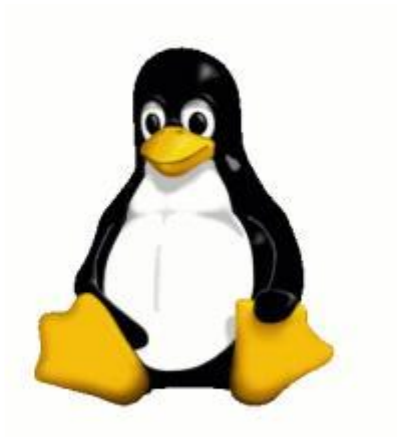
Counter (CTR) mode encryption

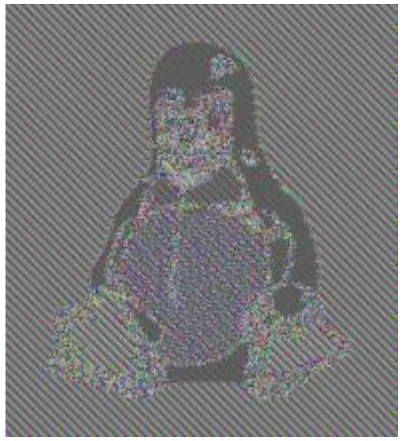
Sources:
1. Wikipedia: Block Cipher mode of operation. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation. Retrieved July 10, 2021.

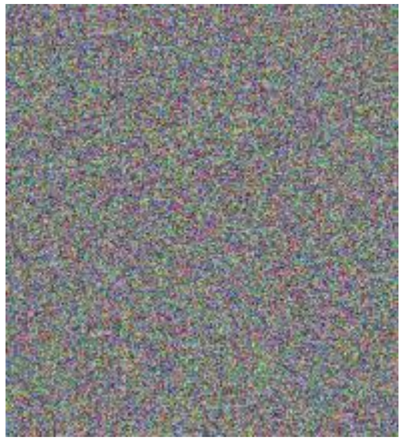
Block based ciphers use nonces and an Initialization Vector (IV)

- **Many cryptographic modes require an “initialization vector” (IV)**
  - Need not be secret, but must be unpredictable
- **Helps the encryption scheme achieve semantic security where repeated use of the scheme with the same key does not allow an attacker to infer relationships between the same encrypted messages.**
- **Some implementations are poor and use 0’s.**
- **Others do not use secure randomly generated IVs or use predictable ones!**
- **Please beware of how nonce and IV are combined and used in the algorithm, otherwise might break security guarantees of the cryptographic algorithm/primitive/protocol!**

ECB vs Other Modes







Original Image

ECB Mode

Other / Desired

Images retrieved from WikiMedia Commons on December 6, 2012. Released to public domain by creator Larry Ewing (lewing@isc.tamu.edu). Imaged created using GIMP.
Credit: Filippo Valsorda. This image was inspired by the original lower-resolution ECB Penguin image by Wikipedia User:Lunkwill. Source "The ECB Penguin" (2013-11-10), https://blog.filippo.io/the-ecb-penguin/; licensed under CC BY-SA 4.0 International. Based on the "Tux the penguin" official Linux mascot created by Larry Ewing in 1996.

Authenticated encryption with block ciphers

- **It is used to try and simultaneously verify the data integrity and the authenticity of the message.**
- **MAC – A message authentication code and usually involves a combination of the following three algorithms:**
  - Key generation algorithm (random)
  - Key based Signing algorithm – returns a signature given key and message
  - Verification algorithm to verify message is neither tampered nor forged.
- **HMAC – Hash-based MAC. Uses a key based hash computation function to compute MAC.**
- **Examples: HMAC-SHA1 or HMAC-SHA256 etc.**
- **Often combined with symmetric key encryption. To HMAC, to encrypt or to do both?**
  - Encrypt-and-MAC = Encrypt(Plaintext) || MAC(Plaintext) – Approximately used by SSH
  - MAC-then-encrypt = Encrypt(MAC(Plaintext) || Plaintext) – Approximately used by TLS.
  - Encrypt-then-MAC = Encrypt(Plaintext) || MAC(Encrypt(Plaintext))
- **A MAC makes assumptions about the underlying hash algorithm and you need to be careful when choosing a MAC scheme to see whether it applies to your particular application.**

Data Encryption Standard (DES)

- **1973: NIST solicited for a DES**
- **1974: NIST second solicitation, IBM responded**
- **1975: Algorithm published**
- **1976: NIST Workshop, adequate “10-15 years”**
- **1977: DES standardized (FIPS)**
- **Has held up relatively well over time**
  - Do need to avoid weak/semi-weak keys
- **Slow in software**
- **Big problem: Key only 56 variable bits**
  - 64 bit key, but every 8th bit is odd parity (beware: may use known key if not given correct parity!)
  - Easily broken with modern computers/hardware.
  - Do *not* use DES for security today!

Triple DES

- **Uses DES 3 times, with 3 keys K1…K3:**
  - ciphertext = E(K3, D(K2, E(K1, plaintext)))
  - plaintext = D(K1, E(K2, D(K3, ciphertext)))
- **Each DES key 56 bits, full key length 3x56=168 bits**
  - Effective key length 112 bits due to a “man in the middle” attack
  - Just like DES, keys must have correct parity & avoid weak keys
- **Historically relatively secure, but slow**
- **Defined, as Triple Data Encryption Algorithm, in:**
  - NIST Special Publication 800-67 Revision 1
  - ISO/IEC 18033-3:2005 Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers
- **Big problem: Block size only 64 bits (same as Blowfish)**
  - Stop using it; attackable with long-lived connections
  - https://sweet32.info/ , CVE-2016-2183 (TLS), CVE-2016-6329 (OpenVPN)

Advanced Encryption Standard (AES)

- **Replaced DES**
  - DES key too short, 3DES too slow & block too short
- **Developed through open international competition run by NIST**
  - Required algorithms & sample implementation
  - Forbid patents (so anyone can use)
  - Competition was a fantastic success
- **15 candidates; Rijndael won & became AES**
  - U.S. FIPS PUB 197
- **A simple-to-understand detailed description is at:**
  - http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

AES (2)

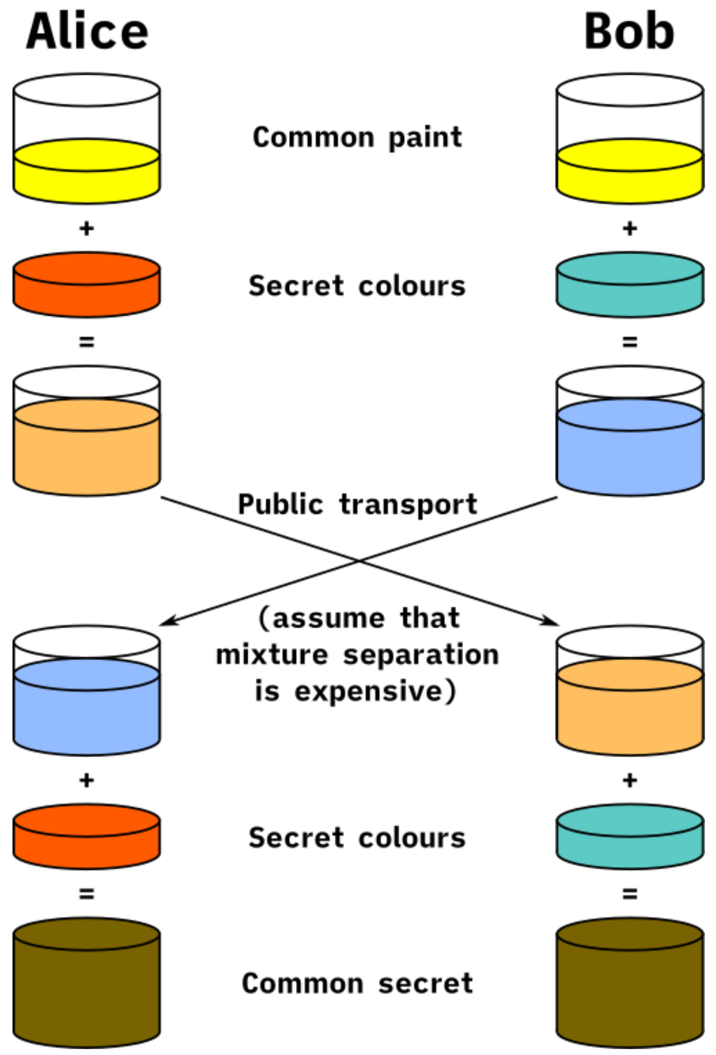
- **Rijndael algorithm allows block and key sizes of any multiple of 32 bits, 128…256 bits**
- **AES is a limited variant of Rijndael with block size=128, key sizes = 128, 192, or 256 bits**
  - Algorithm internally repeats; as key lengthens, number of iterations increases
  - 128bit key: 10 cycles, 192bit: 12, 256bit: 14
- **Much faster than 3DES, longer keys**

Cryptographic Protocols

- **A key part of any information system involves communicating and exchanging that information.**
- **How to do this with security related materials like:**
  - Key material
  - Identity material
- **You use cryptographic protocols to do this.**
- **For key exchange, need to be able to communicate partial information such that both parties can establish a shared secret without exposing it …**
- **For authentication, need to use nonces**
- **Examples: Diffie-Hellman-Merkle key exchange, TLS, Needham-Schroeder, Kerberos etc.**

Diffie-Hellman-Merkle Key exchange

- **The Diffie-Hellman key exchange protocol published by Ralph Merkle in 1976:**
  - Describes how two parties Alice and Bob, without prior knowledge,
  - Jointly establish a shared secret key over an insecure channel.
  - The key can then be used to encrypt subsequent communications using a symmetric key based cipher.
  - This does *not authenticate* the two parties to each other, a man-in-the-middle attacker can listen and pretend to be either party.
- **This is used in TLS ephemeral modes (EDH or DHE) to provide perfect forward secrecy:**
  - Each session generates random public keys without using a deterministic algorithm. Thus, if a session key is compromised, newer and older session keys are not.



Sources:
1. A.J. Van Hinck. Introduction to Public Key Cryptography, pp. 16. https://commons.wikimedia.org/wiki/File:Diffie-Hellman_Key_Exchange.svg. Retrieved July 10, 2021.

Cryptographic Protocols contd…

- **Cryptographic protocols use the idea of a cryptographic nonce:**
  - A single-use random or pseudo-random value used to prevent replaying old communications as part of replay attacks.
  - Used to add freshness of values being communicated.
- **Typically used by authentication cryptographic protocols.**

Curve25519

- **Developed by D. J. Bernstein**
- **Intended to be “a state-of-the-art Diffie-Hellman function”**
  - “Given a user's 32-byte secret key, Curve25519 computes the user's 32-byte public key.”
  - “Given a user's 32-byte secret key and another user's 32-byte public key, Curve25519 computes a 32-byte secret shared by the two users. This secret can then be used to authenticate and encrypt messages between the two users.”
- **Based on elliptic-curve cryptography**
- **EC25519 is a related public-key elliptic-curve signature system**
- **More info: “Curve25519: new Diffie-Hellman speed records” by Daniel J. Bernstein, http://cr.yp.to/ecdh.html.**
- **Open Whipser System’s (makers of Signal) implementation of Curve25519 for .NET: https://github.com/signal-csharp/curve25519-dotnet**

Asymmetric Key based Encryption/Decryption

- Schemes so far have used a common shared key known only to established participants in the protocol.
- Asymmetric Key based schemes use two different “keys” to encrypt and then later decrypt messages.

RSA

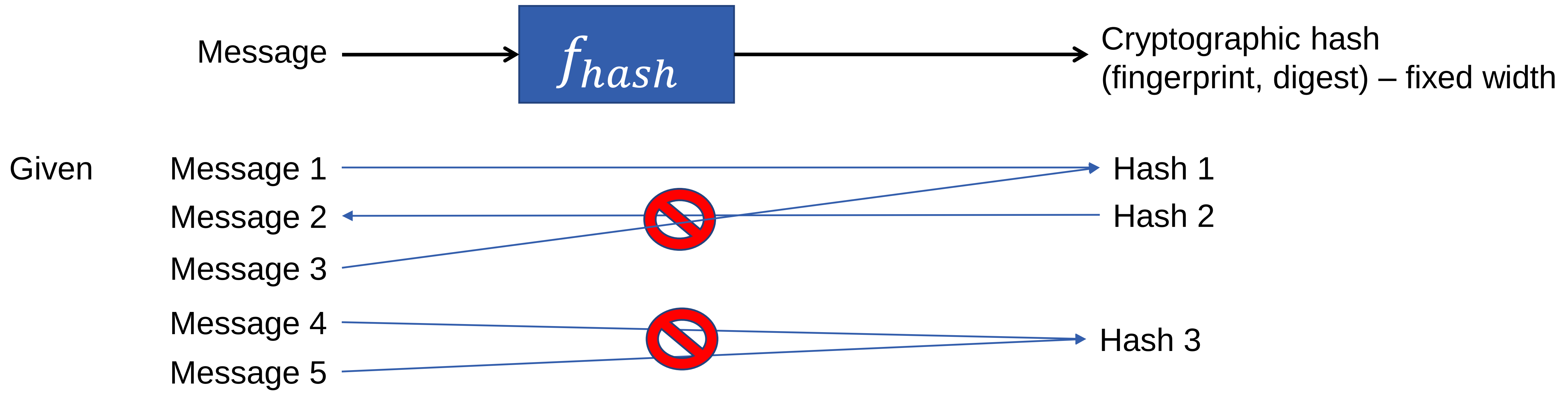
- **Named after Rivest, Shamir, & Adleman**
- **Public key crypto based on difficulty of factoring into prime numbers**
  - Public key & private key
  - Encryption & decryption raise “message” by large exponent
- **Patent released/expired in 2000**
- **Don’t use RSA keys < 1024 bits; 2048+ better**
  - On August 14, 2012, Microsoft issued update to Windows XP & later to block RSA keys <1024 bits
- **Don’t re-implement yourself**
  - The math is easy, doing it correctly for crypto is hard

Asymmetric often used with symmetric algorithms

- **Asymmetric algorithms tend to be slow.**
- **When used to encrypt, often paired with symmetric algorithm:**
  - Idea for sender is to use the asymmetric algorithm to communicate a shared key to receiver, and then use the shared key with symmetric algorithm for subsequent communications.
  - Sender creates single-use “shared” key using cryptographically secure pseudo-random number generator.
  - Shared key encrypted using asymmetric algorithm
  - Receiver receives & decrypts shared key.
  - Rest of data is encrypted with (faster) symmetric algorithm using this single-use key

Cryptographic (one-way) hash function

- **Cryptographic (one-way) hash function takes arbitrary-length data & generates fixed-length hash ("fingerprint") such that it is infeasible to:**
  - Create another message with a given hash value ("pre-image resistance")
  - Create another message with same hash as first message ("second pre-image resistance")
  - Create any two messages with same hash ("collision resistance")



Cryptographic Hash

- **Overall goal: Adversary can’t replace or modify data without changing fingerprint**
- **Some uses:**
  - Verifying integrity – just store fingerprint, verify by recomputing to ensure unchanged/expected
  - Digital signing – Use private key to “encrypt” fingerprint; anyone can use public key to verify
  - File id – e.g., CM systems
  - Creating “random” values
  - Password storage (“per-user salted hashes”)

Cryptographic hash algorithms: MD5, SHA-1, SHA-2

- **MD5: Was widely used, but now broken**
- **SHA-1: Many moved to it, still widely used**
  - Cryptanalysis work in 2004 found important weaknesses
  - Broken: https://shattered.io/
- **SHA-2**
  - Family: SHA-224, SHA-256, SHA-384, SHA-512
  - Technical similarities with SHA-1 raised concerns
- **SHA-3**
  - Released by NIST on August 5, 2015.
  - Part of broader cryptographic primitive family called Keccak designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche that won the NIST competition started on November 2, 2007.

Lifecycles of popular crytographic hashes (Valerie Aurora)

Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
Snefru	🟩	🟩	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD4	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5			🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD2			🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD			🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0				🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
RIPEMD-128							🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
RIPEMD-160							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜
SHA-2 family											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨
SHA-3 (Keccak)																				🟩	🟩	🟩	🟩

In 2004 Xiaoyun Wang et al. published "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD."

RIPEMD-128 is only 128-bit, "irresponsible based on sheer digest length" because at best it's 2^64 complexity to break

Key: 🟩 Unbroken 🟧 Weakened 🟥 Broken ⬜ Deprecated

Source: "Lifecycles of popular crytographic hashes", Valerie Aurora, http://valerieaurora.org/hash.html

Reactions to stages in the life cycle of cryptographic hash functions

Stage	Expert reaction	Programmer reaction	Non-expert ("slashdotter") reaction
Initial proposal	Skepticism, don't recommend use in practice	Wait to hear from the experts before adding to OpenSSL	SHA-what?
Peer reviewal	Moderate effort to find holes and garner an easy publication	Used by a particularly adventurous developers for specific purposes	Name-drop the hash at cocktail parties to impress other geeks
General acceptance	Top-level researchers begin serious work on finding a weakness (and international fame)	Even Microsoft is using the hash function now	Flame anyone who suggests the function may be broken in our lifetime
Minor weakness discovered	Massive downloads of turgid pre-prints from arXiv, calls for new hash functions	Start reviewing other hash functions for replacement	Long semi-mathematical posts comparing the complexity of the attack to the number of protons in the universe
Serious weakness discovered	Tension-filled CRYPTO rump sessions! A full break is considered inevitable	Migrate to new hash functions immediately, where necessary	Point out that no actual collisions have been found
First collision found	Uncork the champagne! Interest in the details of the construction, but no surprise	Gather around a co-worker's computer, comparing the colliding inputs and running the hash function on them	Explain why a simple collision attack is still useless, it's really the second pre-image attack that counts
Meaningful collisions generated on home computer	How adorable! I'm busy trying to break this new hash function, though	Send each other colliding X.509 certificates as pranks	Tell people at parties that you always knew it would be broken
Collisions generated by hand	Memorize as fun party trick for next faculty mixer	Boggle	Try to remember how to do long division by hand
Assumed to be weak but no one bothers to break	No one is getting a publication out of breaking this	What's this crypto library function for?	Update Pokemon Wikipedia pages

Source: "Lifecycles of popular cryptographic hashes", Valerie Aurora, http://valerieaurora.org/hash.html

To learn more

- See Christof Paar’s Cryptography youtube lecture series:
- https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg/videos

Cryptography weaknesses in software systems

CWE-759: Use of a One-Way Hash without a Salt

- **Description: The software uses a one-way cryptographic hash against an input that should not be reversible, such as a password, but the software does not also use a salt as part of the input.**
- **This makes it easier for attackers to pre-compute the hash value using dictionary attack techniques such as rainbow tables.**
- **It should be noted that, despite common perceptions, the use of a good salt with a hash does not sufficiently increase the effort for an attacker who is targeting an individual password, or who has a large amount of computing resources available, such as with cloud-based services or specialized, inexpensive hardware.**
- **Offline password cracking can still be effective if the hash function is not expensive to compute; many cryptographic functions are designed to be efficient and can be vulnerable to attacks using massive computing resources, even if the hash is cryptographically strong.**
- **The use of a salt only slightly increases the computing requirements for an attacker compared to other strategies such as adaptive hash functions.**

Sources:
1. CWE-759: Use of a One-Way Hash without a Salt. MITRE CWE. https://cwe.mitre.org/data/definitions/759.html. Retrieved July 20, 2021.

The Birthday Attack (For Hashes)

- **In Probability Theory, the birthday paradox concerns the probability that in a set of *n* randomly chosen people, some pair of them will have the same birthday.**
- **In a group of:**
  - 23 people, the probability of a shared birthday exceeds 50%.
  - 70 people, there is a 99.9% chance of a shared birthday.
  - 367 people, the probability reaches 100%.
- **The idea is extended to use this probabilistic model to reduce the complexity of finding a collision for a hash function, as well as to calculate the approximate risk of a hash collision existing within the hashes of a given population (for example a list of dictionary words).**

Hash table example

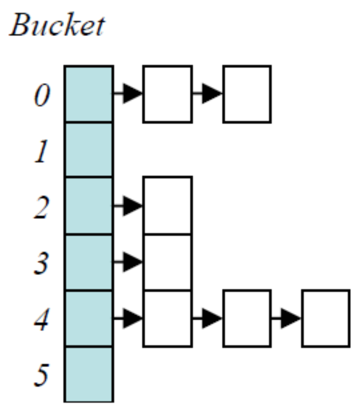


Figure 1: Normal operation of a hash table.

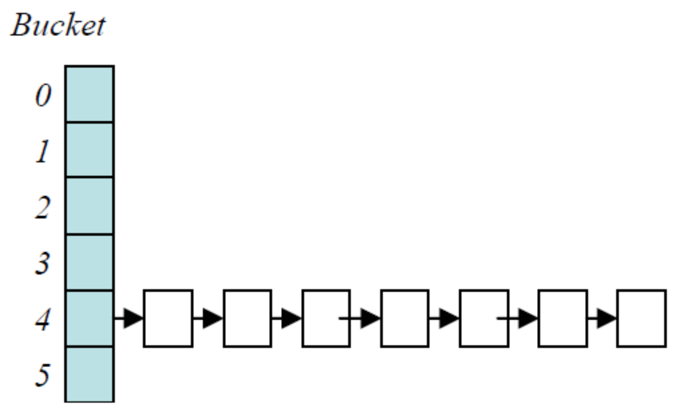


Figure 2: Worst-case hash table collisions.

Source: Crosby, Scott A., and Dan S. Wallach.
Denial of Service via Algorithmic Complexity Attacks.
Usenix Security 2003.

Hash collision impact

- **Klink & Wälde: Not only can you pre-compute, but you can often calculate “backwards” to make attacks especially easy. (Rainbow tables)**
- **One client, 1 Gbit/s connection, can keep busy:**
  - PHP 5: 10,000 i7 cores
  - ASP.NET: 30,000 Core2 cores
  - Java + Tomcat 6.0.32: 100,000 i7 cores
  - Python + Plone: 50,000 Core2 cores
  - Ruby 1.8: 1,000,000 (one million) i7 cores

Source: Klink, Alexander “alech” and Julian “zeri” Wälde. “Efficient Denial of Service Attacks on Web Application Platforms”. December 28th, 2011. 28th Chaos Communication Congress.

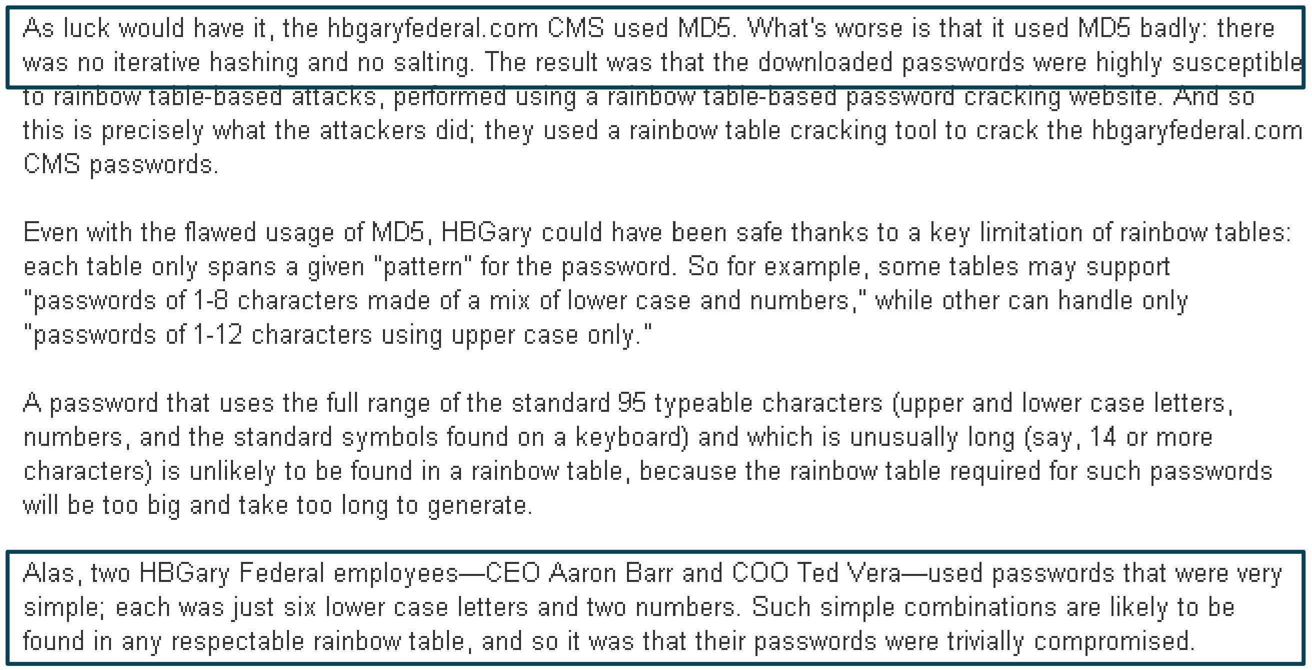
CWE-916: Use of Password Hash With Insufficient Computational Effort

- **Description: The software generates a hash for a password, but it uses a scheme that does not provide a sufficient level of computational effort that would make password cracking attacks infeasible or expensive.**
- **Many password storage mechanisms compute a hash and store the hash, instead of storing the original password in plaintext. In this design, authentication involves accepting an incoming password, computing its hash, and comparing it to the stored hash.**
- **There are several properties of a hash scheme that are relevant to its strength against an offline, massively-parallel attack:**
  - The amount of CPU time required to compute the hash ("stretching")
  - The amount of memory required to compute the hash ("memory-hard" operations)
  - Including a random value, along with the password, as input to the hash computation ("salting")
  - Given a hash, there is no known way of determining an input (e.g., a password) that produces this hash value, other than by guessing possible inputs ("one-way" hashing)
  - Relative to the number of all possible hashes that can be generated by the scheme, there is a low likelihood of producing the same hash for multiple different inputs ("collision resistance")

Sources:
1. CWE-916: Use of Password Hash With Insufficient Computational Effort. MITRE CWE. https://cwe.mitre.org/data/definitions/916.html. Retrieved July 20, 2021.

Real World Example – Hash Compromise

Remember the Anonymous attack discussed earlier?



As luck would have it, the hbgaryfederal.com CMS used MD5. What's worse is that it used MD5 badly: there was no iterative hashing and no salting. The result was that the downloaded passwords were highly susceptible to rainbow table-based attacks, performed using a rainbow table-based password cracking website. And so this is precisely what the attackers did; they used a rainbow table cracking tool to crack the hbgaryfederal.com CMS passwords.

Even with the flawed usage of MD5, HBGary could have been safe thanks to a key limitation of rainbow tables: each table only spans a given "pattern" for the password. So for example, some tables may support "passwords of 1-8 characters made of a mix of lower case and numbers," while other can handle only "passwords of 1-12 characters using upper case only."

A password that uses the full range of the standard 95 typeable characters (upper and lower case letters, numbers, and the standard symbols found on a keyboard) and which is unusually long (say, 14 or more characters) is unlikely to be found in a rainbow table, because the rainbow table required for such passwords will be too big and take too long to generate.

Alas, two HBGary Federal employees—CEO Aaron Barr and COO Ted Vera—used passwords that were very simple; each was just six lower case letters and two numbers. Such simple combinations are likely to be found in any respectable rainbow table, and so it was that their passwords were trivially compromised.

Bright, P. (2011, February 15). *Anonymous speaks: The inside story of the HBGary hack*. Retrieved February 16, 2011, from http://arstechnica.com/tech-policy/news/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack.ars/
Image: From Wikimedia Commons – Image taken by Vincent Diamante, February 10, 2008. Retrieved September 20, 2011 from https://secure.wikimedia.org/wikipedia/commons/wiki/File:Anonymous_at_Scientology_in_Los_Angeles.jpg

Poor choice of an Initialization Vector for Block based cryptography

- **Block based cryptographic algorithms use:**
  - An initialization vector and
  - Potential padding in order to fit the block size
- **Choosing an IV of 0 is a poor choice**

Use what you check (Android "Master Key")

- **CVE-2013-4787 / Android bug 8219321**
  - Reported to Google by Jeff Forristal (Rain Forest Puppy)
  - Packages are really "zip" archive files
  - Special zip files can be created where >1 file can exist in a directory with same name
  - One is checked for its fingerprint… but a different one is actually installed
- **App stores can check for such malformed zip files**

Source:
1. Wikipedia: Android Developer Day. From https://en.wikipedia.org/wiki/Android_Developer_Day. Licensed CC-By-3.0.

CWE-330: Use of Insufficiently Random Values

- **Description: The software uses insufficiently random numbers or values in a security context that depends on unpredictable numbers.**
- **When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.**

Sources:
1. CWE-330: Use of Insufficiently Random Values. MITRE CWE. https://cwe.mitre.org/data/definitions/330.html. Retrieved July 20, 2021.

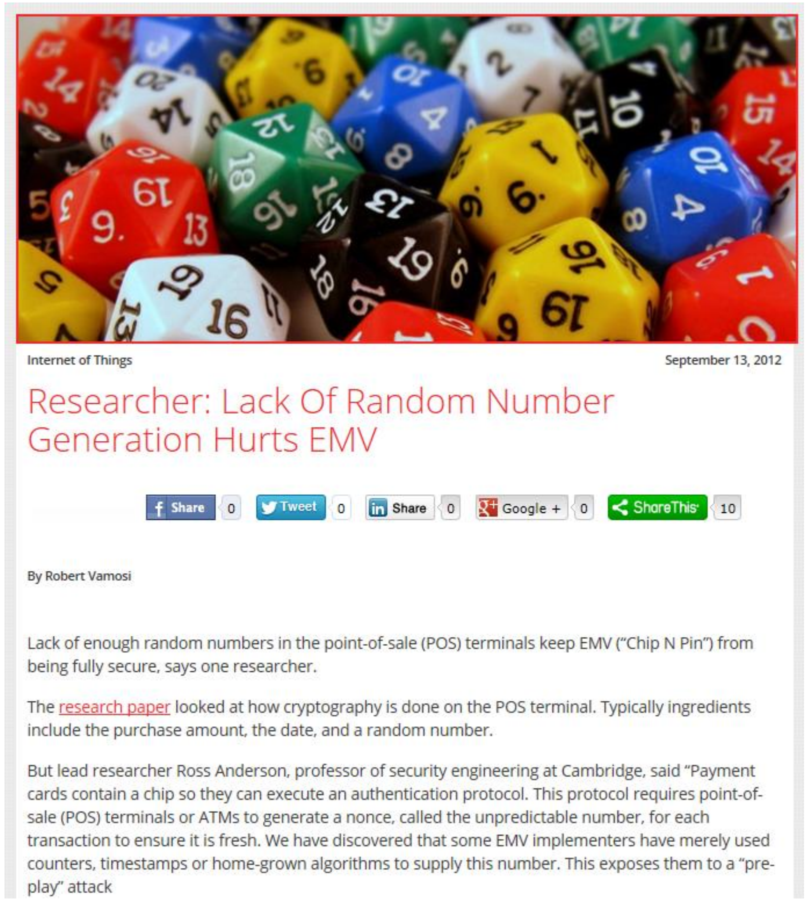
Insecure Random Number Generator

- **A lot of software systems rely on random values as part of their implementation:**
  - Initialization Vectors for cryptographic algorithms
  - Nonces for cryptographic algorithms
  - Statistical sampling
  - Session IDs
- **Some random number generators require seed values to initialize them.**
- **Improper random numbers can lead to predictable values, which can lead to the cryptographic algorithm implementation being broken.**

Real World Example - Chip and Pin

- **Many ATMs and point-of-sale terminals use a predictable random number**

- **Attack:**
  - attacker predicts "unpredictable number" (UN)
  - customer uses a controlled terminal
  - "extra" transaction is performed using the UN and a future date
  - chip on credit card produces an Authorization Request Cryptogram (ARQC) based on UN
  - when time is right, attacker uses fake card with pre-recorded ARQC at ATM to withdraw cash


Internet of Things

September 13, 2012

Researcher: Lack Of Random Number Generation Hurts EMV

By Robert Vamosi

Lack of enough random numbers in the point-of-sale (POS) terminals keep EMV ("Chip N Pin") from being fully secure, says one researcher.

The research paper looked at how cryptography is done on the POS terminal. Typically ingredients include the purchase amount, the date, and a random number.

But lead researcher Ross Anderson, professor of security engineering at Cambridge, said "Payment cards contain a chip so they can execute an authentication protocol. This protocol requires point-of-sale (POS) terminals or ATMs to generate a nonce, called the unpredictable number, for each transaction to ensure it is fresh. We have discovered that some EMV implementers have merely used counters, timestamps or home-grown algorithms to supply this number. This exposes them to a "pre-play" attack

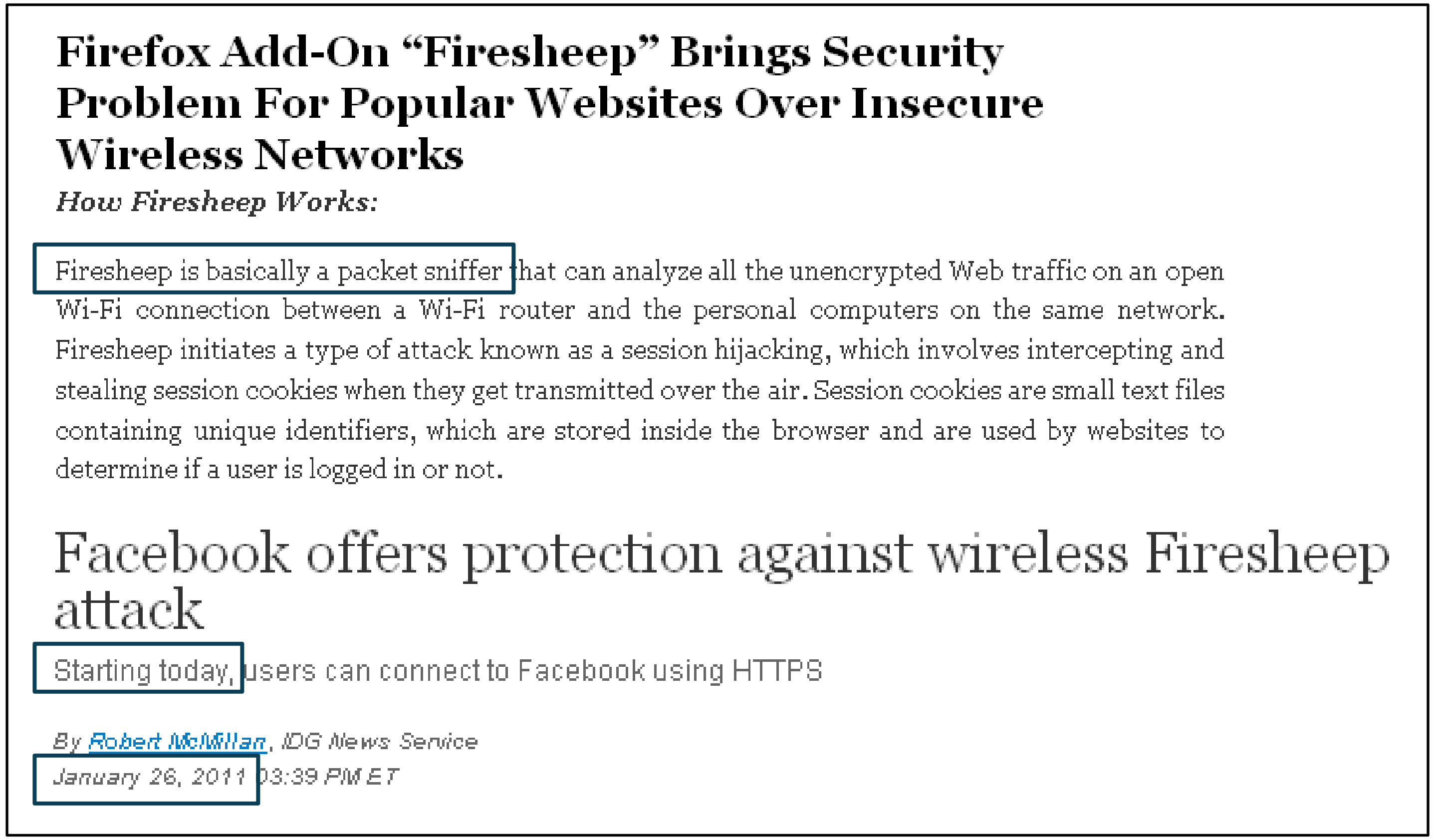
Vamosi, R. (2012, September 13). *Researcher: Lack of Random Number Generation Hurts EMV.* Retrieved November 5, 2014, from http://www.mocana.com/blog/2012/09/13/researcher-lack-of-random-number-generation-hurts-emv

RSA & DH: Some concerns

- **Concerns about RSA & Diffie-Hellman (DH) raised at Black Hat 2013**
  - RSA and DH underpinned by difficulty of "discrete logarithm problem"
  - French academic Antoine Joux published two papers suggesting an algorithm to break it could be found before long
  - "Our conclusion is there is a small but definite chance that RSA and classic Diffie-Hellman will not be usable for encryption purposes in four to five years" - Alex Stamos, chief technology officer, Artemis
  - "The RSA protocol that is the foundation of security on the Internet is likely to be broken in the very near future," Philippe Courtot, CEO of Qualys.
  - http://www.technologyreview.com/news/517781/math-advances-raise-the-prospect-of-an-internet-security-crisis/).
  - Timing side channel attack (David Brumley et al.): https://users.ece.cmu.edu/~dbrumley/courses/18487-f15/reading/Brumley,%20Boneh_2003_Remote%20timing%20attacks%20are%20practical.pdf
- **Elliptic Curve Cryptography (ECC) techniques available**
  - Certicom used to hold patents (since the techniques were created by the company founders), as noted earlier, but they have mostly expired.

Real World Example – Packet Capture

Remember the Firesheep plug-in attack discussed earlier?



Firefox Add-On "Firesheep" Brings Security Problem For Popular Websites Over Insecure Wireless Networks

How Firesheep Works:

Firesheep is basically a packet sniffer that can analyze all the unencrypted Web traffic on an open Wi-Fi connection between a Wi-Fi router and the personal computers on the same network. Firesheep initiates a type of attack known as a session hijacking, which involves intercepting and stealing session cookies when they get transmitted over the air. Session cookies are small text files containing unique identifiers, which are stored inside the browser and are used by websites to determine if a user is logged in or not.

Facebook offers protection against wireless Firesheep attack

Starting today, users can connect to Facebook using HTTPS

By Robert McMillan, IDG News Service
January 26, 2011 03:39 PM ET

Pillai, J. (2010, October 28). *Firefox Add-on "Firesheep" brings security problem for popular websites over insecure wireless networks*. Retrieved February 25, 2011, from http://news.ebrandz.com/miscellaneous/2010/3657-firefox-add-on-firesheep-brings-security-problem-for-popular-websites-over-insecure-wireless-networks-.html
McMillan, R. (2011, January 26). *Facebook offers protection against wireless Firesheep attack*. Retrieved March 3, 2011, from http://www.networkworld.com/news/2011/012611-facebook-offers-protection-against-wireless.html

CWE-523: Unprotected Transport of Credentials

- **Description: Login pages do not use adequate measures to protect the user name and password while they are in transit from the client to the server.**
- **SSL (Secure Socket Layer) provides data confidentiality and integrity to HTTP. By encrypting HTTP messages, SSL protects from attackers eavesdropping or altering message contents.**
- **Not any version of SSL will do. SSL implementations vary and**
- **See https://mitls.org/. See the apps/TLSharp project in mitls-flex repository for .NET example (Written using F#.NET).**

Sources:
1. CWE-523: Unprotected Transport of Credentials. MITRE CWE. https://cwe.mitre.org/data/definitions/523.html. Retrieved July 20, 2021.

Real World Example - POODLE

Encryption is HARD!

- **Vulnerability in CBC encryption in SSL v3.0**
  - SSL has been around for 18 years!
- **Attacker downgrades to SSL v3.0 which uses a MAC-then-encrypt scheme.**
- **The padding bytes appended at end are not authenticated using MAC. The server happily decrypts the last block (padding). Attacker copies the appropriate cipher block (let's say containing cookie values).**
- **Man in the Middle attacker**
  - controls request (using JavaScript)
  - padding fills an entire block
  - reveals one byte at a time

SSL broken, again, in POODLE attack

Yet another flaw could prove to be the final nail in SSLv3's coffin.

by **Peter Bright** - Oct 15 2014, 12:15am EDT

Share Tweet 82

Poodle Gothic Redux by Amanda Wray

Sources:
1. Retrieved October 27, 2014, from http://arstechnica.com/security/2014/10/ssl-broken-again-in-poodle-attack/.
2. Retrieved July 22, 2021 from https://www.openssl.org/~bodo/ssl-poodle.pdf.

Timing attacks: The problem

- **Many algorithms take a variable amount of time**
  - E.G., array “is equal to?” usually stops on first unequal value
- **Attackers can get confidential info (like private keys) through this**
  - Attackers perform act multiple times…
  - Use statistics to eliminate jitter; a shocking amount is removable!
  - E.G., for “is equal to” attacker can determine if he guessed first one correctly, then second one, then third one, …
- **Yes, it really works**
  - 15-100µs accuracy across the Internet and 100ns over a local network [Crosby2009]
  - One example (finding 20 byte HMAC result): Keyczar used standard equality test, allowing attacker to find session value in < week with 10 req/s… “and all of a sudden I’m logged in as you” [Hale2009]
  - If attacker on same system or has shared resources (such as mobile apps or virtualized cloud), *timing attack problems are even worse*

Next time …

- Cryptography related security bugs - Defenses