Queensland University of Technology

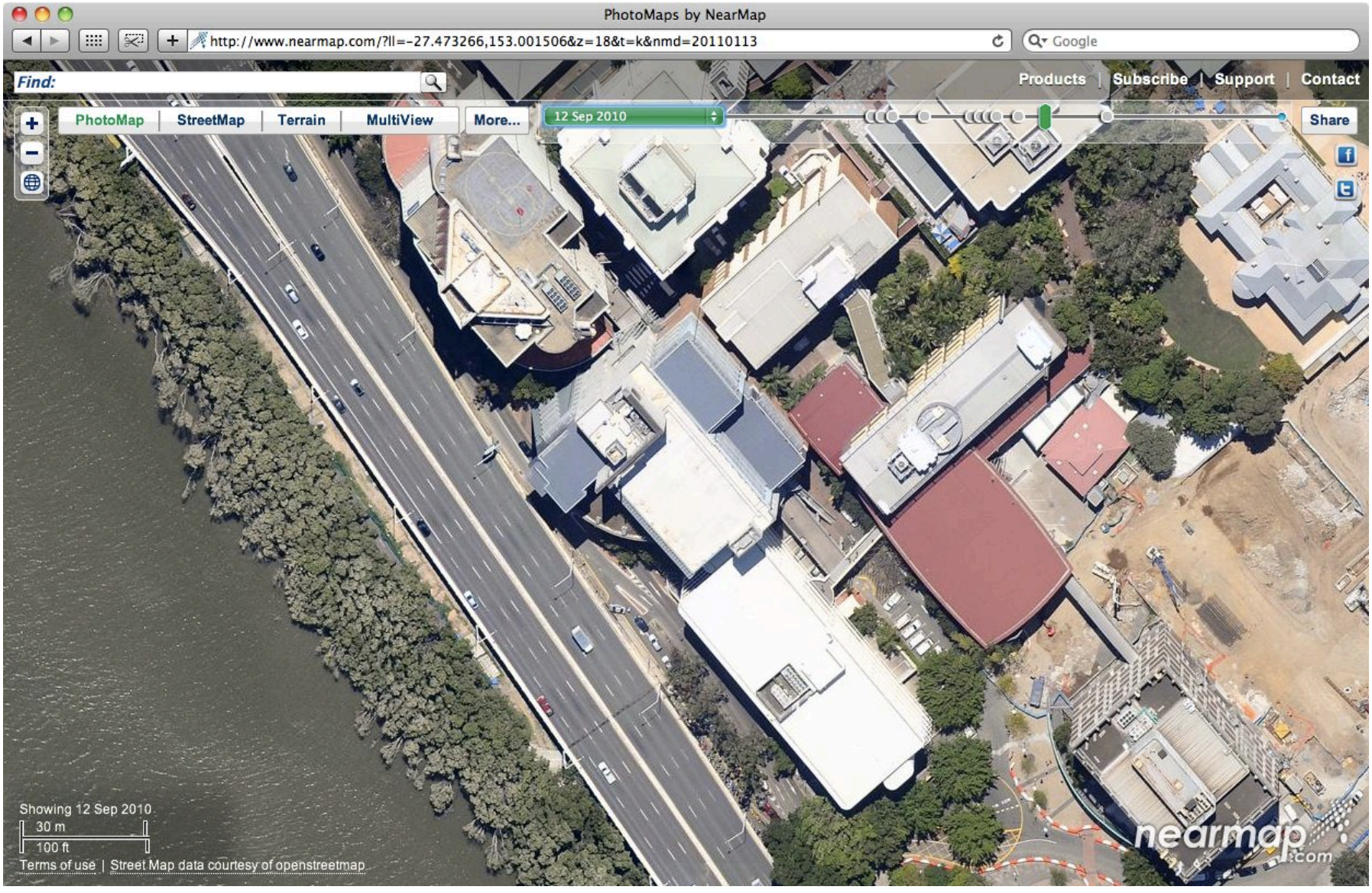
DoS-resistant key exchange: models and mechanisms

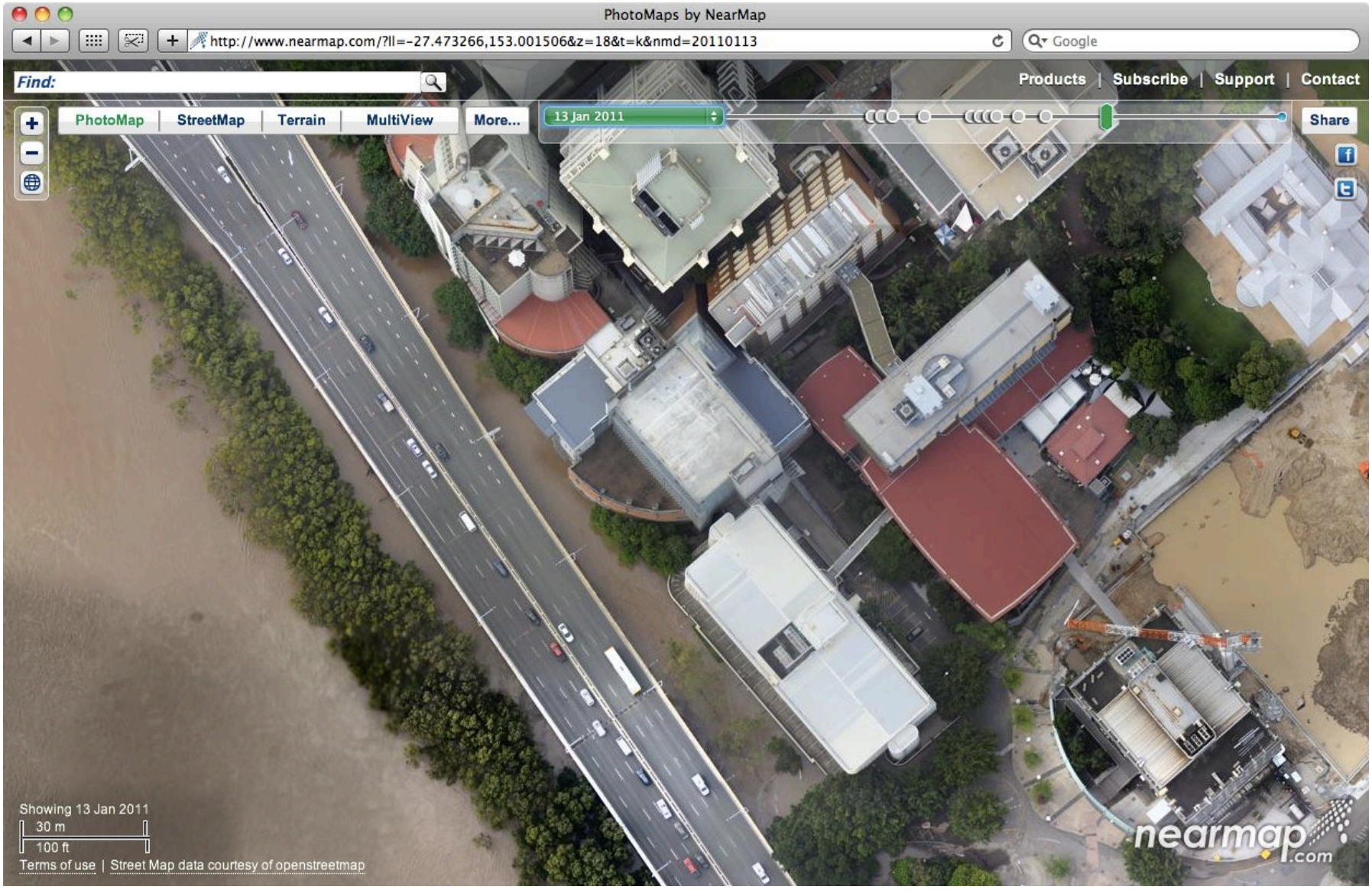
Douglas Stebila

Joint work with Colin Boyd, Juan González-Nieto, Lakshmi Kuppusamy, Jothi Rangasamy

Information Security Institute
Queensland University of Technology, Brisbane, Queensland, Australia

Tuesday, January 25, 2011


PhotoMaps by NearMap
http://www.nearmap.com/?ll=-27.473266,153.001506&z=18&t=k&nmd=20110113
Google
Find:
Products | Subscribe | Support | Contact
PhotoMap
StreetMap
Terrain
MultiView
More...
12 Sep 2010
Share
Showing 12 Sep 2010
30 m
100 ft
Terms of use | Street Map data courtesy of openstreetmap
nearmap.com


PhotoMaps by NearMap
http://www.nearmap.com/?ll=-27.473266,153.001506&z=18&t=k&nmd=20110113
Google
Find:
Products | Subscribe | Support | Contact
PhotoMap | StreetMap | Terrain | MultiView | More...
13 Jan 2011
Share
Showing 13 Jan 2011
30 m
100 ft
Terms of use | Street Map data courtesy of openstreetmap
nearmap.com


Gardens Point

Australia-India Project

This work is part of the Australia-India Strategic Research Fund (AISRF) project on *Protecting Critical Infrastructure from Denial of Service Attacks*.

- Subproject 1: Advanced high-rate packet classifier
- Subproject 2: DoS defences for web services and service-oriented architectures
- **Subproject 3: DoS-resilient authentication protocols**
- Subproject 4: DoS vulnerabilities in emerging technologies
- Subproject 5: Harmonisation of policy, legal and regulatory environments

Outline

Cyber attacks

- **Estonia (April 2007)**: ping flooding, botnets; affected websites of Estonian parliament, banks, ministries, newspapers.
- **Georgia (August 2008)**: hacking/defacement, denial of service; affected media sites.
- **United States and South Korea (July 2009)**: DDoS using botnets (est. 20,000-500,000 computers); affected websites of White House, Pentagon, various South Korean government websites. Allegedly launched by North Korean telecommunications ministry.

Cyber attacks

- **Estonia (April 2007)**: ping flooding, botnets; affected websites of Estonian parliament, banks, ministries, newspapers.
- **Georgia (August 2008)**: hacking/defacement, denial of service; affected media sites.
- **United States and South Korea (July 2009)**: DDoS using botnets (est. 20,000-500,000 computers); affected websites of White House, Pentagon, various South Korean government websites. Allegedly launched by North Korean telecommunications ministry.
- **Google (June 2009)**: News searches sparked by Michael Jackson's death were initially mistook for an automated denial of service attack.

Cyber attacks

BBC Mobile News | Sport | Weather | iPlayer | TV

NEWS TECHNOLOGY

Home | World | UK | England | N. Ireland | Scotland | Wales | Business | Politics | Health | Education | Sci/En

9 December 2010 Last updated at 09:10

Anonymous hacktivists say Wikileaks war to continue

A member of the Anonymous group of hackers, which has been targeting firms it sees as being anti-Wikileaks has said the campaign is not over.

Speaking on the BBC's Today programme, Coldblood said that "more and more people are downloading the voluntary botnet tool".

This signs them up to a so-called botnet, an

Types of denial of service attacks

- **Brute force attacks**: attacker generates sufficiently many legitimate requests to overload a server's resources. Does not require special knowledge of protocol specification or implementation.
  - Distributed denial of service (DDoS) attacks
  - Ping floods
- **Semantic attacks**: attacker tries to exploit vulnerabilities of particular network protocols or applications. Requires special knowledge of protocol specification and implementation.
  - Buffer overflow attacks
  - TCP SYN flooding / IP spoofing attacks

Prevention techniques

Try to identify malicious traffic:

- address filtering to block false addresses or addresses making too many requests;
- bandwidth management by routers and switches;
- packet inspection: look for patterns of bad requests;
- intrusion-prevention systems: look for signatures of attacks.

Difficult to distinguish real users' legitimate requests from attacker's legitimately-formed requests in brute force attacks.

Gradual authentication

- Principle for denial-of-service resistance proposed by Meadows.
- Idea is to use cheap and low-security authentication initially.
- Gradually put more effort into authentication if earlier stages succeed.

Cookies, puzzles and cryptographic authentication

- **Cookies** provide proof of reachability.
- **Puzzles** provide proof of work.
- **Signatures** provide strong cryptographic authentication.

Puzzles

The server generates a challenge and the client is required to solve a moderately hard puzzle based on this challenge.
Puzzles should be:

- easy to generate,
- not require stored state,
- provably hard to solve, and
- easy to verify.

Puzzles may be either **computation-bound** or **memory-bound**. We only look at the former.

Puzzle security properties

- **Difficulty**: it should be moderately hard to solve a puzzle (computation-bound or memory-bound).
- **Unforgeability**: it should not be possible to generate valid puzzles.
- **Non-parallelizability**: it should not be possible to have multiple computers solve a puzzle in less time than a single computer could.
- **Tuneable difficulty**: can provide puzzles with different difficulty levels.
- **Useful puzzles**: the work done in solving a puzzle can be used for another purpose.

Outline

Puzzle definition

Formally, a client puzzle is a tuple of algorithms:

- **Setup**(1^k): Return public parameters and server secret s.
- **GenPuz**(s, d, str): Generate a puzzle of difficulty d for session string str.
- **FindSoln**(str, puz): Find a solution for session string str and the given puzzle puz.
- **VerSoln**(s, str, puz, $soln$) Check if $soln$ is a valid solution for puzzle puz and session string str.

GenPuz and VerSoln should be very inexpensive.

SPuz: puzzle based on Juels–Brainard construction

Client **Server**

$\xrightarrow{\texttt{Req}}$

Choose random $x \leftarrow \{0,1\}^k$

$x = \underbrace{x'}_{Q} \parallel \underbrace{x''}_{k-Q}$

$y = H(x, Q, \texttt{str})$

$\xleftarrow{x'', y}$

Find z such that
$H(z \parallel x'', Q, \texttt{str}) = y$

$\xrightarrow{\texttt{str}, x'', y, z}$

$y \stackrel{?}{=} H(z \parallel x'', Q, \texttt{str})$

Puzzle difficulty: the Bristol definition

- Experiment parameters: puzzle difficulty d, security parameter k, and puzzle scheme P.
- Adversary interacts with a challenger which runs Setup(1^k) and provides access to two oracles:
  - **CreatePuzSoln**(str): Set $puz \leftarrow \text{GenPuz}(s, d, str)$ and find a valid solution *soln* for *puz*. Return (*puz, soln*).
  - **Test**(str^*): Return $puz^* \leftarrow \text{GenPuz}(s, d, str^*)$. Only a single Test query is allowed.
- Goal: output $soln^*$ such that $\text{VerSoln}(puz^*, soln^*)$ is true.

Puzzle difficulty: the Bristol definition

A client puzzle scheme is said to be $\epsilon_{k,d}()$-**difficult** if

$$\Pr(\text{A wins}) \leq \epsilon_{k,d}(t)$$

for all probabilistic algorithms A running in time at most t, where $\epsilon_{k,d}(t)$ is a family of functions monotonically increasing in t.

- Example: might have $\epsilon_{k,d}(t) = t/d + negl(k)$.
- Why monotonically increasing? Should be impossible to solve a puzzle more easily by taking less time.

Limitation in Bristol definition

- Does not address the ability of powerful adversaries to solve multiple puzzles.
- We might choose puzzle difficulty 2^{20} operations, because we want a puzzle that takes a couple of seconds to solve on a modern CPU. There are definitely adversaries that have more power than that, so the puzzle difficulty experiment says nothing about them.
- Can construct examples based on signatures using composite modulus. Puzzle solution is a signature forgery.
  - Forging one signature can be easier than factoring the modulus and so Bristol definition is satisfied.
  - Forging 2^{20} signatures may only take the effort of forging, say, 2^{10} signatures by factoring the modulus and then using the trapdoor.

Strong puzzle difficulty

- We introduce new security experiments to address this weakness (and provide additional functionality).
- Quantify the ability of an adversary to return multiple solutions, not just one.
- The adversary can return solutions $(str, puz, soln)$ where it queried (str, puz) to the puzzle solving oracle, provided $soln$ was not the given solution
- Adversary has access to separate oracles for puzzle generation and puzzle solving.

Strong puzzle difficulty

Goal: output a list of n tuples $(str_i, puz_i, soln_i)$ such that

1. $\text{VerSoln}(s, str_i, puz_i, soln_i)$ is true,
2. (str_i, puz_i) was generated by the puzzle generation oracle, and
3. $soln_i$ was not the response of any puzzle solution query for (str_i, puz_i)

A client puzzle scheme is said to be $\epsilon_{k,d,n}()$-**strongly-difficult** if $\Pr(\text{A wins}) \leq \epsilon_{k,d,n}(t)$ for all probabilistic algorithms A running in time at most t, where

$$\epsilon_{k,d,n}(t) \leq \epsilon_{k,d,1}(t/n)$$

for all t, n such that $\epsilon_{k,d,n}(t) \leq 1$.

SPuz is strongly difficult

Theorem

Let H be a random oracle and let $\epsilon_{k,Q,n}(q) = \left(\frac{q+n}{n2^Q}\right)^n$. Then SPuz is an $\epsilon_{k,Q,n}(q)$ strongly difficult client puzzle, where q is the number of distinct queries to H.

Proof is based on counting number of queries to H. Suppose that q_i is the number of queries used to attempt to solve puzzle i. Then $q = q_1 + \ldots + q_n$ and adversary's success probability is bounded by

$$\prod_{i=1}^{n} \frac{q_i + 1}{2^Q} \leq \left(\frac{\sum_{i=1}^{n}(q_i + 1)}{n2^Q}\right)^n = \left(\frac{q+n}{n2^Q}\right)^n \quad .$$

Defining DoS-resistant protocols

- Adversary controls communication between all parties.
- Adversary can gain server secret information via Expose query.
- Adversary can get clients to solve puzzles.
- The probability that an efficient adversary can make the server accept n puzzle instances should be bounded by a non-decreasing function $\epsilon_{k,n}(t)$ where $\epsilon_{k,n}(t) \leq \epsilon_{k,1}(t/n)$.
- Server should not perform expensive operations in a protocol run until puzzle is solved.

Generic construction using client puzzles

- Easiest way to protect a cryptographic protocol using client puzzles is to prepend the protocol run with a client puzzle run, and only run the main protocol once the puzzle is accepted.
- Let P be a protocol, Puz be a puzzle, and let $D(P, Puz)$ be the protocol in which each run of P is prepended by a run of Puz, protected by a MAC keyed by a server secret.

Theorem (Informal)

If Puz is a strongly difficult puzzle, then $D(P, Puz)$ is a denial-of-service-resistant protocol.

If the first round of P involves expensive operations, then this adds an extra round; otherwise, we may be able to combine message flows.

Simple pre-session construction

Client

Choose random N_C

$\xrightarrow{N_C}$

Server (secret ρ)

Choose random $x \leftarrow \{0,1\}^k$

$x = \underbrace{x'}_{Q} \,||\, \underbrace{x''}_{k-Q}$

$\mathtt{str} = (C, S, N_C, N_S)$

$y = H(x, Q, \mathtt{str})$

$\sigma = MAC_\rho(\mathtt{str}, x'', y)$

$\xleftarrow{N_S, x'', y, \sigma}$

Find z such that
$H(z \,\|\, x'', Q, \mathtt{str}) = y$

$\xrightarrow{\mathtt{str}, x'', y, z, \sigma}$

Check for replay

Verify MAC σ

$y \stackrel{?}{=} H(z \,\|\, x'', Q, \mathtt{str})$

Conclusion (models)

- Provided new stronger definition for puzzle difficulty.
- Showed existence of efficient puzzles satisfying the new definition.
- Paper also includes definition for **non-interactive puzzle difficulty**.
- Showed how to apply puzzles in a generic way to achieve DoS-resistant protocols.

Outline

Aura's puzzle

- Aura, Nikander and Leiwo, 2000.
- Server chooses nonce N_S and difficulty level Q. These are sent to the client.
- Client C generates nonce N_C. Needs to find X so that:

$$H(C, N_S, N_C, X) = \underbrace{00\ldots000}_{Q \text{ bits}} Y$$

  Client C returns X together with nonce N_C.
- Puzzle verification uses only one hash call
- If H is a random function then client needs to make around 2^Q hash function calls before solving the puzzle

Limitations of Aura's puzzle

- **No checking of client reachability**: puzzle does not incorporate cookie-like property based on client identity/address.
- **No solution may exist**: puzzle solution is not generated by server.
- **Granularity is exponential**: can only increase difficulty level by doubling at each step.

A generalised Aura puzzle

- We allow a solution to be value d with $d < D$ for some maximum value D.
- For each time period the server selects a secret K and generates a nonce N_S.
- Server also chooses the puzzle difficulty level Q and an Q-bit integer D.

A generalized Aura puzzle

Client		**Server**
	$\xrightarrow{\text{Req}}$	
		$Z = H_K(N_S, Q, D, C)$
	$\xleftarrow{Z, N_S, Q, D}$	
Generate N_C $M = Z\|\|N_S\|\|N_C\|\|S\|\|C$ Find X such that $H(M, X) \bmod 2^Q \leq D$		
	$\xrightarrow{Z, N_S, Q, D, N_C, X}$	
		Check (N_S, Q, D) recent and N_C not reused $Z \stackrel{?}{=} H_K(N_S, Q, D, C)$ $M = Z\|\|N_S\|\|N_C\|\|S\|\|C$ Check $H(M, X) \bmod 2^Q \leq D$

Properties of new puzzle

- Each puzzle issued is a function of the client identity. Therefore proof of reachability is obtained.
- The puzzle can be proven to be **strongly difficulty** with

$$\epsilon_{k,Q,n}(t) = \left(\frac{(D+1)(q+1)}{n2^Q}\right)^n$$

  where q is the number of calls to the random hash function.
- The granularity is linear in D.

Bernstein's signatures

- Initially proposed by Dan Bernstein in 2002.
- Based on earlier idea by Rabin and Williams.
- The signature of a message is a square root of the (randomised) hash of the message.
- Verification is the fastest for any known signature: requires one modular squaring and one modular multiplication, but can be improved by reducing large values to smaller ones.

Bernstein's signatures

Let H be a hash function. The signature schemes consists of the following algorithms.

- **KeyGen**: Generate an RSA private key $sk = (p, q)$ and corresponding public key $pk = n = pq$.
- **Sign**$(sk = (p, q), m)$: Compute a signature (r, h, f, t, s) such that $h = H(m, r)$ for a random r, $f \in \{-2, -1, 1, 2\}$, and $s^2 = f \cdot h + t \cdot n$.
- **Verify**$(pk = n, m, (r, h, f, t, s))$: Check if $h = H(m, r)$ and

$$s^2 \equiv f \cdot h + t \cdot n \bmod u$$

  where u is a 'secret' prime of around 115 bits.

Integrating puzzle into Bernstein signatures

- The value $H(M, X)$ in the puzzle becomes the hash value $H(m, r)$ used in the Bernstein signature.
- The client solves the puzzle and computes its signature using the hash value $H(M, X)$.
- The server checks the puzzle solution and, if correct, continues to verify signature.
- The puzzle is checked 'for free' since the hash is computed anyway to verify the signature.

Comparing performance of RSA and Bernstein signatures

	verification operations per second OpenSSL v1.0.0 64-bit x86_64 build	
modulus (bits)	RSA ($e = 65537$) with separate puzzle	FVDS (full verify) with built-in puzzle
1024	29630	180938
1536	14891	122558
2048	8710	103962
4096	2354	46532

DoS resistance for SSL

- One of the most widely deployed key exchange protocols.
- Inherently carries no resistance to denial-of-service.
- We implemented a new cipher suite incorporating the new combined puzzle and signature.
- Server must first check puzzle, then verify signature, then perform RSA decryption of pre-master secret.

Number of connections achieved

Configuration	RSA-1024	FVDS-1024
no puzzle	1621	1732
diff= 2^{12}; legitimate solutions	1597	1719
diff= 2^{12}; garbage solutions	3734	4030
diff= 2^{12}; mix legitimate/garbage	100 legitimate 2767 garbage	100 legitimate 3022 garbage

FVDS shows better performance in all cases.

Conclusion (integrated mechanism)

- Designed improved client puzzle.
- Shown the theoretical and practical possibility of combining puzzle and signatures.
- Demonstrated effectiveness of method in SSL.
- Could provide more dramatic improvement given more efficient server-side key exchange.

Further reading

- Chen, Morrissey, Smart, and Warinschi. Security Notions and Generic Constructions for Client Puzzles. *ASIACRYPT 2009*.
- Stebila, Kuppusamy, Rangasamy, Boyd and Gonzalez-Nieto. Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols. *CT-RSA 2011*.
- Rangasamy, Stebila, Boyd and Gonzalez-Nieto. An Integrated Approach to Cryptographic Mitigation of Denial-of-Service Attacks. *ASIACCS 2011*.