

Client puzzles for denial-of-service resistant authentication

Joint work with Juan Gonzalez, Lakshmi Kuppusamy, Jothi Rangasamy, Douglas Stebila, Suriadi Suriadi

Colin Boyd

Information Security Institute
Queensland University of Technology

December 2011

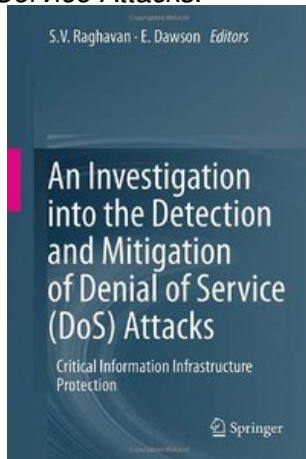
Outline

- 1 Background
 - What is DoS?
 - Defending against DoS
- 2 Types of puzzle
 - Hash-based puzzles
 - Number-theoretic (RSA-based) puzzles
- 3 Computational models for puzzles
- 4 Case study: web services

Australia-India Project

Part of Australia-India Strategic Research Fund project on *Protecting Critical Infrastructure from Denial of Service Attacks*.

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



Cyber attacks

- Denial-of-service (DoS) is one of the most common real world network security attacks.
- DoS prevents users from accessing their legitimate resources. It is an attack on *availability*.
- Highly publicised attacks have affected nation states: Estonia (April 2007); Georgia (August 2008); United States and South Korea (July 2009).
- DoS attacks against sites of your choice are readily available for hire.

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.
- Can authentication help?

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
- A typical progression might be to implement cookies first, then puzzles, then strong 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,
- easy to verify.

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

Puzzle definition

Formally, a client puzzle is a tuple of algorithms:

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

Puzzle security properties

- Difficulty: it should be moderately hard to solve a puzzle
- Unforgeability: it should not be possible for the adversary 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

Hash-based puzzle (Juels–Brainard)

Based on finding partial pre-image of hash function H . Difficulty parameter is Q .

- PuzGen**
- Choose random $x \leftarrow \{0, 1\}^k$
 - Set $x = \underbrace{x'}_Q \parallel \underbrace{x''}_{k-Q}$
 - Set $z = H(x, Q, \text{str})$
 - Puzzle is (x'', z)

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

VerSoln Check that $z \stackrel{?}{=} H(y \parallel x'', Q, \text{str})$

Properties of hash-based puzzles

Merits

- Generation and verification very efficient
- Easily tuneable by giving 'hints' (range for solution)

Limitations

- Seem hard to make non-parallelisable
- Proofs of difficulty are only available in the random oracle model

Time-lock puzzles of Rivest–Shamir–Wagner (RSW)

- RSA-based puzzle proposed in 1996
- *Sending information into the future*
- Uses RSA modulus $n = pq$.

Setup

- Choose difficulty Q
- Compute $b = 2^Q \bmod \phi(n)$

PuzGen

- Choose random x
- Puzzle consists of (n, x, Q)

FindSoln

- Compute $y = x^{2^Q} \bmod n$

VerSoln

- Check that $y \stackrel{?}{=} x^b \bmod n$

Properties of RSW puzzle

Merits

- Believed to be non-parallelisable - only known way to find y is to square a repeatedly Q times.
- Simple construction

Limitations

- Verification requires exponentiation
- No proof of difficulty

Karame–Čapkun puzzle (ESORICS 2010)

- RSW puzzle is relatively expensive to verify. VerSoln requires full modular exponentiation.
- Karame and Čapkun use *short RSA private exponent*. Consequently RSA public exponent must be very large.
- Puzzle is essentially to compute RSA encryption of random value.
- Verification is decryption with short exponent and checking.

Karame–Čapkun construction

n is RSA modulus, d is short RSA private exponent of length k (such as $k = 80$), public exponent is $e > n^2$.

- Setup**
- Choose difficulty Q
 - Compute $b = 2^Q \bmod \phi(n)$

- PuzGen**
- Choose random X
 - $K = e - b$
 - Puzzle is (n, x, Q, K)

FindSoln Compute $y_1 = x^{2^Q} \bmod n$; $y_2 = x^K \bmod n$

VerSoln Check that $(y_1 y_2)^d \bmod n \stackrel{?}{=} x$

Properties of Karame–Čapkun construction

Merits

- Verification much improved over RSW puzzle, by about $|n|/2k$ times
- Has proof of difficulty (relative to RSW puzzle)

Limitations

- Verification still requires exponentiation
- Parallelisability not so tight

BPV generator

- Boyko, Peinado, Venkatesan, Eurocrypt'98
- Method for computing random RSA encryptions and exponentiations efficiently with pre-computation.
- Pre-computation generates a table of random pairs:
 - $\{(\alpha_i, \alpha_i^U)\}$ for RSA generator;
 - $\{(x_i, g^{x_i})\}$ for DL generator;
- When new value is needed a small random subset of table is chosen and combined.
- For suitable parameter sizes the output of the generators are statistically indistinguishable from randomly generated pairs.

BPV generator for RSA

BPV Generator

Let k , ℓ , and N , with $N \geq \ell \geq 1$, be parameters. Let n be an RSA modulus and u an exponent.

- **Pre-processing** run once. Generate N random integers $\alpha_1, \alpha_2, \dots, \alpha_N \leftarrow \mathbb{Z}_n^*$ and compute $\beta_i \leftarrow \alpha_i^u \bmod n$ for each i . Return a table $\tau \leftarrow ((\alpha_i, \beta_i))_{i=1}^N$.
- **Whenever a pair $(x, x^u \bmod n)$ is needed:** choose a random set $S \subseteq \{1, \dots, N\}$ of size ℓ . Compute $x \leftarrow \prod_{j \in S} \alpha_j \bmod n$ and $X \leftarrow \prod_{j \in S} \beta_j \bmod n$ and return (x, X) .

A new non-parallelisable puzzle (RSA Puz)

n is RSA modulus, public exponent is $e = 3$.

- Setup**
- Set $d = 3^{-1} \bmod \phi(n)$
 - Choose difficulty Q
 - Set $u = d - (2^Q \bmod \phi(n))$
 - Compute BPV pre-processing to obtain table with $N = 2500$ and $\ell = 4$

- PuzGen**
- Use BPV algorithm to computer new $(x, X = x^u)$ pair
 - Puzzle is (n, x, Q)

FindSoln Compute $y = x^{2^Q} \bmod n$

VerSoln Check that $(X \cdot y)^3 \bmod n \stackrel{?}{=} x$

Properties of RSA Puz

Merits

- Verification only requires a few multiplications
- Non-parallelisable
- Has proof of difficulty (relative to RSW puzzle)

Limitations

- Preprocessing can be somewhat costly

A new puzzle with difficulty provable in the standard model (DL Puz)

Setup ● Choose random b and compute $g^b \bmod n$.

- Computer BPV preprocessing to obtain table of (x_i, g^{x_i}) values.

PuzGen ● Use BPV algorithm to computer new (a, g^a) pair

- Choose random z
- Set $v = a + bz \bmod \phi(n)$
- Set l to be random interval of length Q containing v .
- Puzzle is (n, g, z, g^a, g^b, l)

FindSoln ● Compute $V = g^a \cdot (g^b)^z \bmod n$

- Find v with $g^v \bmod n = V$.

VerSoln Check that $v \stackrel{?}{=} a + bz \bmod \phi(n)$

Properties of DL Puz

Merits

- Verification only requires only one multiplication
- Has proof of difficulty in standard model

Limitations

- Preprocessing can be somewhat costly
- Parallelisable

Sample timings

Puzzle	512-bit modulus, $k = 56$			
	Setup (ms)	GenPuz (μ s)	FindSoln (s)	VerSoln (μ s)
Difficulty: $Q = 10^6$ ($\times 10$ for DL puz)				
RSW puz	13.92	4.80	1.54	474.68
KC puz	11.52	8.37	1.59	263.35
RSA puz	1401.11	16.66	1.54	14.75
DL puz	31.86	31.43	1.05	5.31
Difficulty: $Q = 10^7$ ($\times 15$ for DL puz)				
RSW puz	49.99	4.80	15.17	474.83
KC puz	28.95	8.37	15.18	265.28
RSA puz	1419.78	16.66	15.34	14.53
DL puz	31.83	32.01	18.10	5.29
Difficulty: $Q = 10^8$ ($\times 15$ for DL puz)				
RSW puz	416.29	4.81	157.10	470.61
KC puz	218.76	8.35	160.97	259.39
RSA puz	1609.83	16.76	158.22	14.88
DL puz	31.89	32.01	175.41	5.27

Puzzle difficulty: the Bristol definition (Asiacrypt 2009)

- Adversary is allowed to see valid puzzles and solutions.
- Adversary asks for a challenge puzzle and has to solve it.
- A puzzle scheme is said to be $\epsilon_{k,Q}()$ -difficult if

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

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

- Example: might have $\epsilon_{k,d}(t) = t/d + \text{negl}(k)$.

Strong puzzle difficulty (CT-RSA 2011)

- New security definition to address the ability of powerful adversaries to solve multiple puzzles.
- Adversary has access to separate oracles for puzzle generation and puzzle solving.
- Adversary goal is to output n valid puzzles and solutions for which it was not previously given solutions.

Strong puzzle difficulty

A client puzzle scheme is said to be $\epsilon_{k,d,n}()$ -strongly-difficult if $\Pr(A \text{ wins}) \leq \epsilon_{k,d,n}(t)$ 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 $\epsilon_{k,d,n}(t) \leq 1$.

Difficulty proofs for existing puzzles

- Only puzzles so far known to satisfy strong definition is hash-based puzzle using random oracle.
- Both Karame–Čapkun puzzle (KC Puz) and our RSA-based puzzle (RSA Puz) satisfy Bristol definition if RSW puzzle is difficult
- Our discrete logarithm based puzzle (DL Puz) is secure in Bristol definition without random oracles on the assumption of the *modular composite interval discrete logarithm assumption (MIDL)*, a new intractability assumption. MIDL is hard if factorisation and composite IDL problem are both hard.

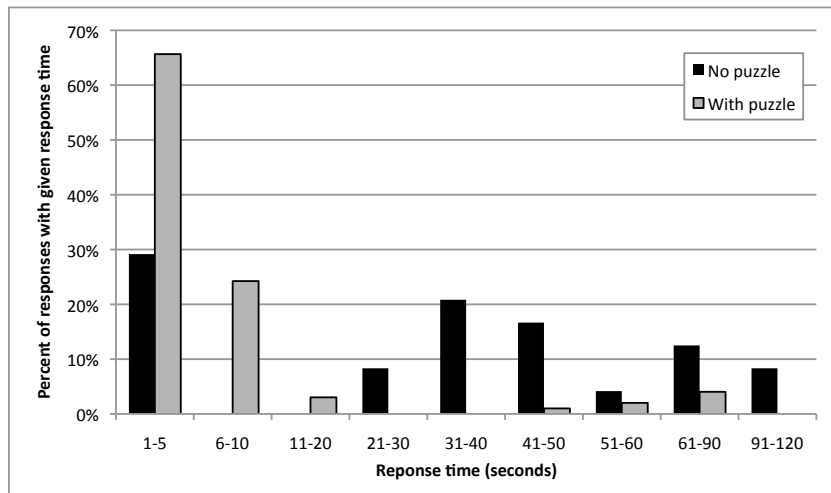
Case study: web services (ICWS 2011)

- Hash-based puzzle integrated into web services implementation
- Various experiments made to assess impact of puzzle on performance
- Examined both flooding and semantic attacks
- For flooding attacks large numbers of requests are made to a service performing a mathematical calculation taking around 84 ms to complete
- For semantic attack small number of requests with huge payload including thousands of digital signatures.
- Always assumed that attacker would not solve puzzle

Flooding attack with mixed traffic

Puzzle	Total Batches	Type	Rate (req/s)	Success Requests	Server Status
No	10	Honest	25	111 (44.4%)	Stall
	10	Malicious	100		
Yes	10	Honest	25	250 (100%)	Alive
	10	Malicious	100		
Yes	30	Honest	25	750 (100%)	Alive
	30	Malicious	100		
Yes	30	Honest	25	748 (99.7%)	Alive
	30	Malicious	150		
Yes	30	Honest	25	299 (39.8%)	Stall
	30	Malicious	180		
Yes	30	Honest	15	449 (99.7%)	Alive
	30	Malicious	180		

Semantic attack with mixed traffic



Conclusion

- Client puzzles provide a demonstrable benefit in tackling an important real world problem.
- A number of practical constructions exist. Properties such as non-parallelizability can also be provided if needed.
- Formal models for defining and proving puzzle security are available.
- Possible further work:
 - find a puzzle with strong difficulty in the standard model;
 - formally measuring work of server.

Further reading

- Rangasamy, Stebila, Boyd and Gonzalez-Nieto An Integrated Approach to Cryptographic Mitigation of Denial-of-Service Attacks, ASIACCS 2011
- Stebila, Kuppusamy, Rangasamy, Boyd and Gonzalez-Nieto. Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols, CT-RSA 2011
- Suriadi, Stebila, Clark, Liu. Defending web services against denial of service attacks using client puzzles, ICWS 2011
- Kuppusamy, Gonzalez-Nieto, Boyd, Stebila and Rangasamy. Efficient Modular Exponentiation-based Puzzles for Denial-of-Service Protection, ICISC 2011
- Kuppusamy, Rangasamy, Stebila, Boyd, Nieto. Towards provably secure DoS-resilient key exchange protocol with perfect forward secrecy, Indocrypt 2011