The Underhanded Crypto(graphy) Contest

2018 Winners

Adam Caudill
AppSec Consulting

Taylor Hornby
University of Waterloo

About Us

- 5 years of Underhanded Crypto! (since 2014)
- Every year, we award prizes to the best new cryptography backdoors.
- Why?
  - **Awareness:** Just because it uses AES-256 doesn't mean it's secure.
  - **Defense:** Backdoors rely on an element of secrecy/undetectability. As we become aware of more and more "backdoor techniques", we learn what to look for and become harder to fool.
  - **Vulnerability Invention:** Some backdoors are "intentional vulnerabilities." We enable researchers to invent new types of crypto vulnerabilities *before* they appear in the wild.

https://underhandedcrypto.com/

2018 Recap

- 3 high-quality submissions and two prizes.

$1,500 worth of Zcash from the Zcash Foundation.

$750 from NCC Group Cryptography Services.

- Thanks to JP Aumasson for judging!

3rd Place: James Bofh, Empty Password Encryption

```
#!/usr/bin/env bash
# Use: ./encrypt-multi.sh file1 file2 ...
PASSWORDLIST=$(mktemp)
while [[ $# -gt 0 ]]; do
    FILENAME="$1"; shift
    openssl rand -base64 32 >>"${PASSWORDLIST}"
    readarray PASS <"${PASSWORDLIST}"
    openssl aes-256-cbc -pass "pass:${PASS[${#PASS[@]}]}" \
      -a -salt -in "${FILENAME}" -out "${FILENAME##*/}.enc"
done
cat "${PASSWORDLIST}"
rm "${PASSWORDLIST}"
```

2nd Place: Ella Rose, Backdoored Key Agreement

Alice:

- Private key: random 256-bit “x”.
- Public key: ex + r mod n.
- Computes the key:

  ms256b(x(ey + z)) = ms256b(exy + xz)

Bob:

- Private key: random 256-bit “y”.
- Public key: ey + z mod n.
- Computes the key:

  ms256b(y(ex + r)) = ms256b(exy + yr)

e = (2048 bits)
265349226705250820975351275557274653240432157519793429713604409839306557887649256474108738783603375627808417670
576555929229898188573533615477244760476701179913702827016832135176031014993424219590568928891491592409549477976
649671921463114377721949022712559075725036030818995489069391338377502199319002855338403831278077586451339382168
881631568770070370926341717529452129405725933534886152712796938052359672488867992016222005573890003285681139114
666638993843920968617878336403722696493355028504423301228332335029847150681141884039292019903042156705645556210
08016057012877571262328589944264490528975008710387975972789065

n = (2048 bits)
543629145155362485514220334471645388520759874291258278762000256607900247709194689998329013604777135375479815641
512985607397763915681151380077221423709280486098633221267915705392252494738834803631921384951888044736478818831
195862796678763655312657625573028952651654153223925650768985762601614859241866580384430608615977767972997640752
439157084639620404166473368162537667668121041611493987569963467689132767934041230365228383413196042042426328214
310209326897477671792546185519011037284817646033923704687191255522514550299170044906797830907096895736993638830
64603411012833624591792761519795347754982143133135049337000177

The Backdoor

Alice:

- Private key: random 256-bit “x”.
- Public key: ex + r mod n.
- Computes the key:

  ms256b(x(ey + z)) = ms256b(exy + xz) (mod n)

Bob:

- Private key: random 256-bit “y”.
- Public key: ey + z mod n.
- Computes the key:

  ms256b(y(ex + r)) = ms256b(exy + yr) (mod n)

Backdoor idea #1: If we choose ‘e’ so that it has a small multiplicative inverse ‘d’ (mod n), where x < d, then computing **d(ex + r) mod n = x + dr** (given x + dr < n), and we learn x + dr. Then **(x + dr) mod d = x**, so we have Alice’s private key.

Backdoor idea #2: That’s too easy, anyone can invert ‘e’ and use the backdoor. Instead, we can pick a special value ‘k’ and make ‘e’ have a small multiplicative inverse **mod n-k**, instead. It turns out that:

- Because ‘k’ is not too big, it’s still true that d(ex+r) = x + dr (mod n-k).
- Now, given only n and e, finding e’s inverse modulo n-k is *as hard as breaking RSA,* so only someone who knows n, e, and k can use the backdoor!

1st Place Winner: Matt Cheung, Weak Curves

- Common practice: use a standard curve for your elliptic curve cryptography.
- What if you want your protocol to negotiate new elliptic curves?
- The protocol will need to verify the curve is secure for doing ECC.
- A *necessary* check is to make sure the curve has a large prime-order subgroup. (Indeed, the backdoored protocol performs this check)
- But that's *not enough*, you also need to check for other kinds of curve weakness, e.g. to the Smart attack.
- Unless you're really up to speed on modern ECC attacks, the small-subgroup check might convince you the protocol is really verifying the curve is secure.

Future

- After 5 years of entries (32 total), can we draw some conclusions?
- If you missed participating this year, stay tuned for the 2019 contest!

contact@underhandedcrypto.com