TweetNaCl: A crypto library in 100 tweets

Daniel J. Bernstein, Bernard van Gastel, Wesley Janssen, Tanja Lange,
Peter Schwabe, Sjaak Smetsers

September 18, 2014

Latincrypt 2014, Florianópolis, Brazil

. . . about two years ago

. . . about two years ago

The security impact of a new cryptographic library

- Networking and Cryptography library (NaCl)
- Easy-to-use, high-level API
  - `crypto_box` for public-key authenticated encryption
  - `crypto_box_open` for verification and decryption
  - `crypto_sign` to generate signed message
  - `crypto_sign_open` to verify signature and recover message

. . . about two years ago

The security impact of a new cryptographic library

- Networking and Cryptography library (NaCl)
- Easy-to-use, high-level API
  - `crypto_box` for public-key authenticated encryption
  - `crypto_box_open` for verification and decryption
  - `crypto_sign` to generate signed message
  - `crypto_sign_open` to verify signature and recover message
- High security
  - Only ≥ 128-bit-secure crypto
  - No secret branch conditions, no secretly indexed memory access
  - No padding oracles
  - Avoid randomness where possible; centralize randomness

. . . about two years ago

The security impact of a new cryptographic library

- Networking and Cryptography library (NaCl)
- Easy-to-use, high-level API
  - `crypto_box` for public-key authenticated encryption
  - `crypto_box_open` for verification and decryption
  - `crypto_sign` to generate signed message
  - `crypto_sign_open` to verify signature and recover message
- High security
  - Only ≥ 128-bit-secure crypto
  - No secret branch conditions, no secretly indexed memory access
  - No padding oracles
  - Avoid randomness where possible; centralize randomness
- High speed
  - Even public-key crypto keeps up with typical network throughput
  - Highly optimized assembly implementations for common platforms

NaCl users

- OpenDNS
- Textsecure
- Tox
- Threema
- QuickTun
- DNSCrypt
- Ethos
- CurveCP
- MinimaLT
- Bittorrent Live
- ZeroMQ

Rather, the problem was that you had to use libraries. If your developer has hit the point where s/he's willing to copy and paste RC4 from Wikipedia, you're already in a kind of Fifth Dimension of laziness. Nobody's going to pull in NaCl or OpenSSL just to encrypt one little blob of text. —Matthew D. Green, July 2013

NaCl features revisited

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✗

NaCl features revisited

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✗
- Easily auditable ✗

Auditability

Auditability

- Code audits of crypto code need expert knowledge
- Code audits are expensive:
  - Cost for TrueCrypt audit: US$50,000
  - Estimated cost for OpenSSL audit: US$250,000

Auditability

- Code audits of crypto code need expert knowledge
- Code audits are expensive:
  - Cost for TrueCrypt audit: US$$50,000$
  - Estimated cost for OpenSSL audit: US$$250,000$
- NaCl code base is not excessive (7569 LOC in C and 19669 LOC in ASM)
- Auditing NaCl is already serious effort
- Ed25519 signatures are waiting to be included in NaCl since 2011
- Ed25519 alone has 5521 LOC in C and 16184 LOC in ASM

Auditability

- Code audits of crypto code need expert knowledge
- Code audits are expensive:
    - Cost for TrueCrypt audit: US$$50,000$
    - Estimated cost for OpenSSL audit: US$$250,000$
- NaCl code base is not excessive (7569 LOC in C and 19669 LOC in ASM)
- Auditing NaCl is already serious effort
- Ed25519 signatures are waiting to be included in NaCl since 2011
- Ed25519 alone has 5521 LOC in C and 16184 LOC in ASM
- Partial audits of Ed25519 found a bug which is triggered with probability $\approx 2^{-60}$

The ideal world

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

The ideal world

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

How to achieve this?

- Short, concise, easy-to-read high-level source code turned into high-speed side-channel-attack-protected machine code by compiler

The ideal world

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

How to achieve this?

- Short, concise, easy-to-read high-level source code turned into high-speed side-channel-attack-protected machine code by compiler
- Fully automated formal verification ensures correctness of source code and compilation process

The ideal world

- High usability ✓
- High security ✓
- High speed ✓
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

How to achieve this?

- Short, concise, easy-to-read high-level source code turned into high-speed side-channel-attack-protected machine code by compiler ✗
- Fully automated formal verification ensures correctness of source code and compilation process ✗
- Current state of the art of compilers and formal verification is quite far from this

Introducing TweetNaCl

- High usability ✓
- High security ✓
- High speed ✗
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

Introducing TweetNaCl

- High usability ✓
- High security ✓
- High speed ✗
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓

The challenge

- Matt Green, Jan. 2013: Is it possible to squeeze a high-security crypto library into 100 tweets?
- Sounds like a fun challenge, but should not make this a code-obfuscation project

Introducing TweetNaCl

- High usability ✓
- High security ✓
- High speed ✗
- High laziness (copy-paste compatible) ✓
- Easily auditable ✓
- High compatibility ✓

The challenge

- Matt Green, Jan. 2013: Is it possible to squeeze a high-security crypto library into 100 tweets?
- Sounds like a fun challenge, but should not make this a code-obfuscation project
- Can we have a **concise** reimplementation of NaCl in 100 tweets?
- Can we do this in C?

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
```

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
crypto_box_beforenm
crypto_box_afternm
crypto_box_open_afternm
```

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
crypto_box_beforenm
crypto_box_afternm
crypto_box_open_afternm
crypto_secretbox = crypto_secretbox_xsalsa20poly1305
crypto_secretbox_open
```

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
crypto_box_beforenm
crypto_box_afternm
crypto_box_open_afternm
crypto_secretbox = crypto_secretbox_xsalsa20poly1305
crypto_secretbox_open
crypto_stream = crypto_stream_xsalsa20
crypto_stream_xor
crypto_stream_salsa20
crypto_stream_salsa20_xor
crypto_core_salsa20
crypto_core_hsalsa20
crypto_onetimeauth = crypto_onetimeauth_poly1305
crypto_onetimeauth_verify
crypto_verify_16
crypto_verify_32
```

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
crypto_box_beforenm
crypto_box_afternm
crypto_box_open_afternm
crypto_secretbox = crypto_secretbox_xsalsa20poly1305
crypto_secretbox_open
crypto_stream = crypto_stream_xsalsa20
crypto_stream_xor
crypto_stream_salsa20
crypto_stream_salsa20_xor
crypto_core_salsa20
crypto_core_hsalsa20
crypto_onetimeauth = crypto_onetimeauth_poly1305
crypto_onetimeauth_verify
crypto_verify_16
crypto_verify_32
crypto_hashblocks = crypto_hashblocks_sha512
crypto_hash = crypto_hash_sha512
```

The NaCl API – a closer look

```
crypto_box = crypto_box_curve25519xsalsa20poly1305
crypto_box_open
crypto_box_keypair
crypto_box_beforenm
crypto_box_afternm
crypto_box_open_afternm
crypto_secretbox = crypto_secretbox_xsalsa20poly1305
crypto_secretbox_open
crypto_stream = crypto_stream_xsalsa20
crypto_stream_xor
crypto_stream_salsa20
crypto_stream_salsa20_xor
crypto_core_salsa20
crypto_core_hsalsa20
crypto_onetimeauth = crypto_onetimeauth_poly1305
crypto_onetimeauth_verify
crypto_verify_16
crypto_verify_32
crypto_hashblocks = crypto_hashblocks_sha512
crypto_hash = crypto_hash_sha512
crypto_scalarmult = crypto_scalarmult_curve25519
crypto_scalarmult_base
crypto_sign = crypto_sign_ed25519
crypto_sign_open
crypto_sign_keypair
```

Reducing code – identifying the modules

- One function for `crypto_stream` and `crypto_stream_xor`
- `crypto_core_salsa20` and `crypto_core_hsalsa20` as wrappers around a single core function

Reducing code – identifying the modules

- One function for `crypto_stream` and `crypto_stream_xor`
- `crypto_core_salsa20` and `crypto_core_hsalsa20` as wrappers around a single `core` function
- Use $\mathbb{F}_{2^{255}-19}$ arithmetic for both Curve25519 and Ed25519
- *Different* scalar multiplication for Curve25519 and Ed25519
- Use complete addition formulas for Ed25519
- Ladder for Ed25519 scalar mult in keygen, signing, and verification

Getting started: #defines and typedefs

- No external #include (minimal codebase)
- Does use external randombytes function
- Only very few #defines and typedefs:

```
#include "tweetnacl.h"
#define FOR(i,n) for (i = 0;i < n;++i)
#define sv static void

typedef unsigned char u8;
typedef unsigned long u32;
typedef unsigned long long u64;
typedef long long i64;
typedef i64 gf[16];
```

Getting started: `#defines` and `typedefs`

- No external `#include` (minimal codebase)
- Does use external `randombytes` function
- Only very few `#defines` and `typedefs`:

```
#include "tweetnacl.h"
#define FOR(i,n) for (i = 0;i < n;++i)
#define sv static void

typedef unsigned char u8;
typedef unsigned long u32;
typedef unsigned long long u64;
typedef long long i64;
typedef i64 gf[16];
```

- Assumption: `u8` has 8 bits
- Assumption: `u64`/`i64` has 64 bits; `i64` is using two's complement

Getting started: `#defines` and `typedefs`

- No external `#include` (minimal codebase)
- Does use external `randombytes` function
- Only very few `#defines` and `typedefs`:

```
#include "tweetnacl.h"
#define FOR(i,n) for (i = 0;i < n;++i)
#define sv static void

typedef unsigned char u8;
typedef unsigned long u32;
typedef unsigned long long u64;
typedef long long i64;
typedef i64 gf[16];
```

- Assumption: `u8` has 8 bits
- Assumption: `u64`/`i64` has 64 bits; `i64` is using two's complement
- Assumption: `u32` has **at least** 32 bits

A glimpse of the code: $\mathbb{F}_{2^{255}-19}$ arithmetic

```
typedef i64 gf[16];

sv A(gf o,const gf a,const gf b)
{
  int i;
  FOR(i,16) o[i]=a[i]+b[i];
}

sv Z(gf o,const gf a,const gf b)
{
  int i;
  FOR(i,16) o[i]=a[i]-b[i];
}
```

```
sv M(gf o,const gf a,const gf b)
{
  i64 i,j,t[31];
  FOR(i,31) t[i]=0;
  FOR(i,16) FOR(j,16) t[i+j]+=a[i]*b[j];
  FOR(i,15) t[i]+=38*t[i+16];
  FOR(i,16) o[i]=t[i];
  car25519(o);
  car25519(o);
}

sv S(gf o,const gf a)
{
  M(o,a,a);
}
```

. . . ctd.

```
sv car25519(gf o)
{
  int i;
  i64 c;
  FOR(i,16) {
    o[i]+=(1LL<<16);
    c=o[i]>>16;
    o[(i+1)*(i<15)]+=c-1+37*(c-1)*(i==15);
    o[i]-=c<<16;
  }
}

sv inv25519(gf o,const gf i)
{
  gf c;
  int a;
  FOR(a,16) c[a]=i[a];
  for(a=253;a>=0;a--) {
    S(c,c);
    if(a!=2&&a!=4) M(c,c,i);
  }
  FOR(a,16) o[a]=c[a];
}
```

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
    - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
    - No global variables
    - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
    - No file handling

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
  - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
  - No global variables
  - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
  - No file handling
- Machine-verified: no array-out-of-bounds access for inputs with certain length requirements

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
    - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
    - No global variables
    - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
    - No file handling
- Machine-verified: no array-out-of-bounds access for inputs with certain length requirements
- Machine-verified: No use of uninitialized data

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
  - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
  - No global variables
  - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
  - No file handling
- Machine-verified: no array-out-of-bounds access for inputs with certain length requirements
- Machine-verified: No use of uninitialized data
- Progress towards integer range-checks (no overflows)

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
  - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
  - No global variables
  - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
  - No file handling
- Machine-verified: no array-out-of-bounds access for inputs with certain length requirements
- Machine-verified: No use of uninitialized data
- Progress towards integer range-checks (no overflows)
- Verification uses C++ overloading

Is TweetNaCl audited yet?

Partially

- Many typical sources for bugs are eliminated by design:
  - No dynamic memory allocation (`malloc`, `free`, `sbrk`, etc.)
  - No global variables
  - No non-bounds-checking functions (`strcpy`, `sprintf`, `sscanf`, etc.)
  - No file handling
- Machine-verified: no array-out-of-bounds access for inputs with certain length requirements
- Machine-verified: No use of uninitialized data
- Progress towards integer range-checks (no overflows)
- Verification uses C++ overloading
- Obviously, TweetNaCl also passes all tests of the NaCl test battery

Are you serious?

Are you serious?

Yes, we are.

- TweetNaCl is timing-attack protected
- TweetNaCl has a really small TCB
- TweetNaCl is truly portable (on one A4 sheet)
- TweetNaCl is auditable (and partially audited)
- TweetNaCl is fast

Are you serious?

Yes, we are.

- TweetNaCl is timing-attack protected
- TweetNaCl has a really small TCB
- TweetNaCl is truly portable (on one A4 sheet)
- TweetNaCl is auditable (and partially audited)
- TweetNaCl is fast enough, for many applications

Are you serious?

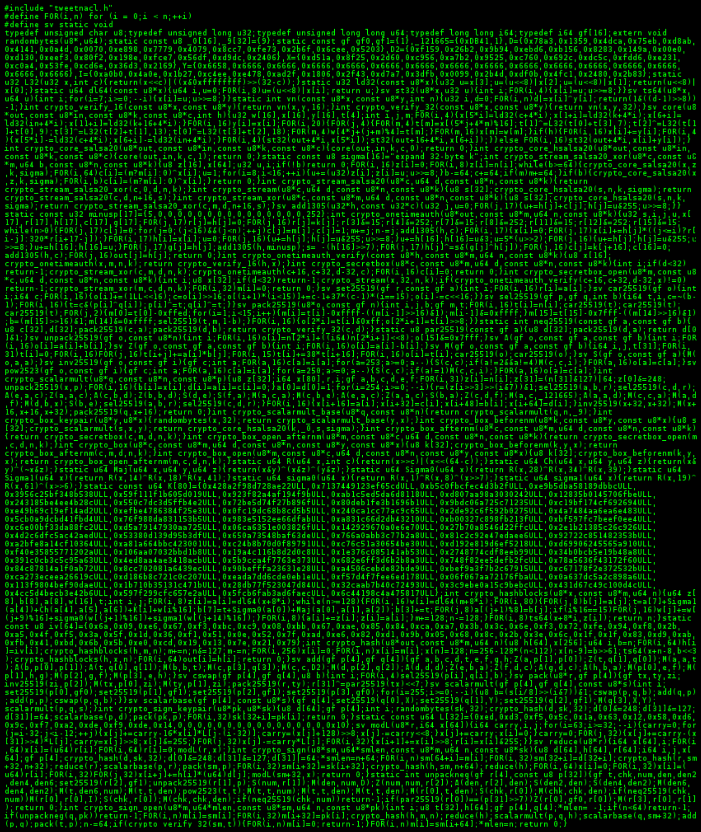
Yes, we are.

- TweetNaCl is timing-attack protected
- TweetNaCl has a really small TCB
- TweetNaCl is truly portable (on one A4 sheet)
- TweetNaCl is auditable (and partially audited)
- TweetNaCl is fast enough, for many applications
- TweetNaCl is compatible to NaCl: Need more speed? Switch to NaCl.

Are you serious?

Yes, we are.

- TweetNaCl is timing-attack protected
- TweetNaCl has a really small TCB
- TweetNaCl is truly portable (on one A4 sheet)
- TweetNaCl is auditable (and partially audited)
- TweetNaCl is fast enough, for many applications
- TweetNaCl is compatible to NaCl: Need more speed? Switch to NaCl.
- TweetNaCl ports/bindings for JS, Ruby, D, Android NDK, Python



```c
#include "tweetnacl.h"
#define FOR(i,n) for (i = 0;i < n;++i)
#define sv static void
typedef unsigned char u8;typedef unsigned long u32;typedef unsigned long long u64;typedef long long i64;typedef i64 gf[16];extern void
randombytes(u8*,u64);static const u8 _0[16],_9[32]={9};static const gf gf0,gf1={1},_121665={0xDB41,1},D={0x78a3,0x1359,0x4dca,0x75eb,0xd8ab,
0x4141,0x0a4d,0x0070,0xe898,0x7779,0x4079,0x8cc7,0xfe73,0x2b6f,0x6cee,0x5203},D2={0xf159,0x26b2,0x9b94,0xebd6,0xb156,0x8283,0x149a,0x00e0,
0xd130,0xeef3,0x80f2,0x198e,0xfce7,0x56df,0xd9dc,0x2406},X={0xd51a,0x8f25,0x2d60,0xc956,0xa7b2,0x9525,0xc760,0x692c,0xdc5c,0xfdd6,0xe231,
0xc0a4,0x53fe,0xcd6e,0x36d3,0x2169},Y={0x6658,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,
0x6666,0x6666},I={0xa0b0,0x4a0e,0x1b27,0xc4ee,0xe478,0xad2f,0x1806,0x2f43,0xd7a7,0x3dfb,0x0099,0x2b4d,0xdf0b,0x4fc1,0x2480,0x2b83};static
u32 L32(u32 x,int c){return(x<<c)|((x&0xffffffff)>>(32-c));}static u32 ld32(const u8*x){u32 u=x[3];u=(u<<8)|x[2];u=(u<<8)|x[1];return(u<<8)|
x[0];}static u64 dl64(const u8*x){u64 i,u=0;FOR(i,8)u=(u<<8)|x[i];return u;}sv st32(u8*x,u32 u){int i;FOR(i,4){x[i]=u;u>>=8;}}sv ts64(u8*x,
u64 u){int i;for(i=7;i>=0;--i){x[i]=u;u>>=8;}}static int vn(const u8*x,const u8*y,int n){u32 i,d=0;FOR(i,n)d|=x[i]^y[i];return(1&((d-1)>>8))
-1;}int crypto_verify_16(const u8*x,const u8*y){return vn(x,y,16);}int crypto_verify_32(const u8*x,const u8*y){return vn(x,y,32);}sv core(u8
*out,const u8*in,const u8*k,const u8*c,int h){u32 w[16],x[16],y[16],t[4];int i,j,m;FOR(i,4){x[5*i]=ld32(c+4*i);x[1+i]=ld32(k+4*i);x[6+i]=
ld32(in+4*i);x[11+i]=ld32(k+16+4*i);}FOR(i,16)y[i]=x[i];FOR(i,20){FOR(j,4){FOR(m,4)t[m]=x[(5*j+4*m)%16];t[1]^=L32(t[0]+t[3],7);t[2]^=L32(t[1
]+t[0],9);t[3]^=L32(t[2]+t[1],13);t[0]^=L32(t[3]+t[2],18);FOR(m,4)w[4*j+(j+m)%4]=t[m];}FOR(m,16)x[m]=w[m];}if(h){FOR(i,16)x[i]+=y[i];FOR(i,4
){x[5*i]-=ld32(c+4*i);x[6+i]-=ld32(in+4*i);}FOR(i,4){st32(out+4*i,x[5*i]);st32(out+16+4*i,x[6+i]);}}else FOR(i,16)st32(out+4*i,x[i]+y[i]);}
int crypto_core_salsa20(u8*out,const u8*in,const u8*k,const u8*c){core(out,in,k,c,0);return 0;}int crypto_core_hsalsa20(u8*out,const u8*in,
const u8*k,const u8*c){core(out,in,k,c,1);return 0;}static const u8 sigma[16]="expand 32-byte k";int crypto_stream_salsa20_xor(u8*c,const u8
*m,u64 b,const u8*n,const u8*k){u8 z[16],x[64];u32 u,i;if(!b)return 0;FOR(i,16)z[i]=0;FOR(i,8)z[i]=n[i];while(b>=64){crypto_core_salsa20(x,z
,k,sigma);FOR(i,64)c[i]=(m?m[i]:0)^x[i];u=1;for(i=8;i<16;++i){u+=(u32)z[i];z[i]=u;u>>=8;}b-=64;c+=64;if(m)m+=64;}if(b){crypto_core_salsa20(x
,z,k,sigma);FOR(i,b)c[i]=(m?m[i]:0)^x[i];}return 0;}int crypto_stream_salsa20(u8*c,u64 d,const u8*n,const u8*k){return
crypto_stream_salsa20_xor(c,0,d,n,k);}int crypto_stream(u8*c,u64 d,const u8*n,const u8*k){u8 s[32];crypto_core_hsalsa20(s,n,k,sigma);return
crypto_stream_salsa20(c,d,n+16,s);}int crypto_stream_xor(u8*c,const u8*m,u64 d,const u8*n,const u8*k){u8 s[32];crypto_core_hsalsa20(s,n,k,
sigma);return crypto_stream_salsa20_xor(c,m,d,n+16,s);}sv add1305(u32*h,const u32*c){u32 j,u=0;FOR(j,17){u+=h[j]+c[j];h[j]=u&255;u>>=8;}}
static const u32 minusp[17]={5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,252};int crypto_onetimeauth(u8*out,const u8*m,u64 n,const u8*k){u32 s,i,j,u,x[
17],r[17],h[17],c[17],g[17];FOR(j,17)r[j]=h[j]=0;FOR(j,16)r[j]=k[j];r[3]&=15;r[4]&=252;r[7]&=15;r[8]&=252;r[11]&=15;r[12]&=252;r[15]&=15;
while(n>0){FOR(j,17)c[j]=0;for(j=0;(j<16)&&(j<n);++j)c[j]=m[j];c[j]=1;m+=j;n-=j;add1305(h,c);FOR(i,17){x[i]=0;FOR(j,17)x[i]+=h[j]*((j<=i)?r[
i-j]:320*r[i+17-j]);}FOR(i,17)h[i]=x[i];u=0;FOR(j,16){u+=h[j];h[j]=u&255;u>>=8;}u+=h[16];h[16]=u&3;u=5*(u>>2);FOR(j,16){u+=h[j];h[j]=u&255;u
>>=8;}u+=h[16];h[16]=u;}FOR(j,17)g[j]=h[j];add1305(h,minusp);s= -(h[16]>>7);FOR(j,17)h[j]^=s&(g[j]^h[j]);FOR(j,16)c[j]=k[j+16];c[16]=0;
add1305(h,c);FOR(j,16)out[j]=h[j];return 0;}int crypto_onetimeauth_verify(const u8*h,const u8*m,u64 n,const u8*k){u8 x[16];
crypto_onetimeauth(x,m,n,k);return crypto_verify_16(h,x);}int crypto_secretbox(u8*c,const u8*m,u64 d,const u8*n,const u8*k){int i;if(d<32)
return-1;crypto_stream_xor(c,m,d,n,k);crypto_onetimeauth(c+16,c+32,d-32,c);FOR(i,16)c[i]=0;return 0;}int crypto_secretbox_open(u8*m,const u8
*c,u64 d,const u8*n,const u8*k){int i;u8 x[32];if(d<32)return-1;crypto_stream(x,32,n,k);if(crypto_onetimeauth_verify(c+16,c+32,d-32,x)!=0)
return-1;crypto_stream_xor(m,c,d,n,k);FOR(i,32)m[i]=0;return 0;}sv set25519(gf r,const gf a){int i;FOR(i,16)r[i]=a[i];}sv car25519(gf o){int
i;i64 c;FOR(i,16){o[i]+=(1LL<<16);c=o[i]>>16;o[(i+1)*(i<15)]+=c-1+37*(c-1)*(i==15);o[i]-=c<<16;}}sv sel25519(gf p,gf q,int b){i64 t,i,c=~(b-
1);FOR(i,16){t=c&(p[i]^q[i]);p[i]^=t;q[i]^=t;}}sv pack25519(u8*o,const gf n){int i,j,b;gf m,t;FOR(i,16)t[i]=n[i];car25519(t);car25519(t);
car25519(t);FOR(j,2){m[0]=t[0]-0xffed;for(i=1;i<15;i++){m[i]=t[i]-0xffff-((m[i-1]>>16)&1);m[i-1]&=0xffff;}m[15]=t[15]-0x7fff-((m[14]>>16)&1)
;b=(m[15]>>16)&1;m[14]&=0xffff;sel25519(t,m,1-b);}FOR(i,16){o[2*i]=t[i]&0xff;o[2*i+1]=t[i]>>8;}}static int neq25519(const gf a,const gf b){
u8 c[32],d[32];pack25519(c,a);pack25519(d,b);return crypto_verify_32(c,d);}static u8 par25519(const gf a){u8 d[32];pack25519(d,a);return d[0
]&1;}sv unpack25519(gf o,const u8*n){int i;FOR(i,16)o[i]=n[2*i]+((i64)n[2*i+1]<<8);o[15]&=0x7fff;}sv A(gf o,const gf a,const gf b){int i;FOR
(i,16)o[i]=a[i]+b[i];}sv Z(gf o,const gf a,const gf b){int i;FOR(i,16)o[i]=a[i]-b[i];}sv M(gf o,const gf a,const gf b){i64 i,j,t[31];FOR(i,
31)t[i]=0;FOR(i,16)FOR(j,16)t[i+j]+=a[i]*b[j];FOR(i,15)t[i]+=38*t[i+16];FOR(i,16)o[i]=t[i];car25519(o);car25519(o);}sv S(gf o,const gf a){M(
o,a,a);}sv inv25519(gf o,const gf i){gf c;int a;FOR(a,16)c[a]=i[a];for(a=253;a>=0;a--){S(c,c);if(a!=2&&a!=4)M(c,c,i);}FOR(a,16)o[a]=c[a];}sv
pow2523(gf o,const gf i){gf c;int a;FOR(a,16)c[a]=i[a];for(a=250;a>=0;a--){S(c,c);if(a!=1)M(c,c,i);}FOR(a,16)o[a]=c[a];}int
crypto_scalarmult(u8*q,const u8*n,const u8*p){u8 z[32];i64 x[80],r,i;gf a,b,c,d,e,f;FOR(i,31)z[i]=n[i];z[31]=(n[31]&127)|64;z[0]&=248;
unpack25519(x,p);FOR(i,16){b[i]=x[i];d[i]=a[i]=c[i]=0;}a[0]=d[0]=1;for(i=254;i>=0;--i){r=(z[i>>3]>>(i&7))&1;sel25519(a,b,r);sel25519(c,d,r);
A(e,a,c);Z(a,a,c);A(c,b,d);Z(b,b,d);S(d,e);S(f,a);M(a,c,a);M(c,b,e);A(e,a,c);Z(a,a,c);S(b,a);Z(c,d,f);M(a,c,_121665);A(a,a,d);M(c,c,a);M(a,d
,f);M(d,b,x);S(b,e);sel25519(a,b,r);sel25519(c,d,r);}FOR(i,16){x[i+16]=a[i];x[i+32]=c[i];x[i+48]=b[i];x[i+64]=d[i];}inv25519(x+32,x+32);M(x+
16,x+16,x+32);pack25519(q,x+16);return 0;}int crypto_scalarmult_base(u8*q,const u8*n){return crypto_scalarmult(q,n,_9);}int
crypto_box_keypair(u8*y,u8*x){randombytes(x,32);return crypto_scalarmult_base(y,x);}int crypto_box_beforenm(u8*k,const u8*y,const u8*x){u8 s
[32];crypto_scalarmult(s,x,y);return crypto_core_hsalsa20(k,_0,s,sigma);}int crypto_box_afternm(u8*c,const u8*m,u64 d,const u8*n,const u8*k)
{return crypto_secretbox(c,m,d,n,k);}int crypto_box_open_afternm(u8*m,const u8*c,u64 d,const u8*n,const u8*k){return crypto_secretbox_open(m
,c,d,n,k);}int crypto_box(u8*c,const u8*m,u64 d,const u8*n,const u8*y,const u8*x){u8 k[32];crypto_box_beforenm(k,y,x);return
crypto_box_afternm(c,m,d,n,k);}int crypto_box_open(u8*m,const u8*c,u64 d,const u8*n,const u8*y,const u8*x){u8 k[32];crypto_box_beforenm(k,y,
x);return crypto_box_open_afternm(m,c,d,n,k);}static u64 R(u64 x,int c){return(x>>c)|(x<<(64-c));}static u64 Ch(u64 x,u64 y,u64 z){return(x&
y)^(~x&z);}static u64 Maj(u64 x,u64 y,u64 z){return(x&y)^(x&z)^(y&z);}static u64 Sigma0(u64 x){return R(x,28)^R(x,34)^R(x,39);}static u64
Sigma1(u64 x){return R(x,14)^R(x,18)^R(x,41);}static u64 sigma0(u64 x){return R(x,1)^R(x,8)^(x>>7);}static u64 sigma1(u64 x){return R(x,19)^
R(x,61)^(x>>6);}static const u64 K[80]={0x428a2f98d728ae22ULL,0x7137449123ef65cdULL,0xb5c0fbcfec4d3b2fULL,0xe9b5dba58189dbbcULL,
0x3956c25bf348b538ULL,0x59f111f1b605d019ULL,0x923f82a4af194f9bULL,0xab1c5ed5da6d8118ULL,0xd807aa98a3030242ULL,0x12835b0145706fbeULL,
0x243185be4ee4b28cULL,0x550c7dc3d5ffb4e2ULL,0x72be5d74f27b896fULL,0x80deb1fe3b1696b1ULL,0x9bdc06a725c71235ULL,0xc19bf174cf692694ULL,
0xe49b69c19ef14ad2ULL,0xefbe4786384f25e3ULL,0x0fc19dc68b8cd5b5ULL,0x240ca1cc77ac9c65ULL,0x2de92c6f592b0275ULL,0x4a7484aa6ea6e483ULL,
0x5cb0a9dcbd41fbd4ULL,0x76f988da831153b5ULL,0x983e5152ee66dfabULL,0xa831c66d2db43210ULL,0xb00327c898fb213fULL,0xbf597fc7beef0ee4ULL,
0xc6e00bf33da88fc2ULL,0xd5a79147930aa725ULL,0x06ca6351e003826fULL,0x142929670a0e6e70ULL,0x27b70a8546d22ffcULL,0x2e1b21385c26c926ULL,
0x4d2c6dfc5ac42aedULL,0x53380d139d95b3dfULL,0x650a73548baf63deULL,0x766a0abb3c77b2a8ULL,0x81c2c92e47edaee6ULL,0x92722c851482353bULL,
0xa2bfe8a14cf10364ULL,0xa81a664bbc423001ULL,0xc24b8b70d0f89791ULL,0xc76c51a30654be30ULL,0xd192e819d6ef5218ULL,0xd69906245565a910ULL,
0xf40e35855771202aULL,0x106aa07032bbd1b8ULL,0x19a4c116b8d2d0c8ULL,0x1e376c085141ab53ULL,0x2748774cdf8eeb99ULL,0x34b0bcb5e19b48a8ULL,
0x391c0cb3c5c95a63ULL,0x4ed8aa4ae3418acbULL,0x5b9cca4f7763e373ULL,0x682e6ff3d6b2b8a3ULL,0x748f82ee5defb2fcULL,0x78a5636f43172f60ULL,
0x84c87814a1f0ab72ULL,0x8cc702081a6439ecULL,0x90befffa23631e28ULL,0xa4506cebde82bde9ULL,0xbef9a3f7b2c67915ULL,0xc67178f2e372532bULL,
0xca273eceea26619cULL,0xd186b8c721c0c207ULL,0xeada7dd6cde0eb1eULL,0xf57d4f7fee6ed178ULL,0x06f067aa72176fbaULL,0x0a637dc5a2c898a6ULL,
0x113f9804bef90daeULL,0x1b710b35131c471bULL,0x28db77f523047d84ULL,0x32caab7b40c72493ULL,0x3c9ebe0a15c9bebcULL,0x431d67c49c100d4cULL,
0x4cc5d4becb3e42b6ULL,0x597f299cfc657e2aULL,0x5fcb6fab3ad6faecULL,0x6c44198c4a475817ULL};int crypto_hashblocks(u8*x,const u8*m,u64 n){u64 z[
8],b[8],a[8],w[16],t;int i,j;FOR(i,8)z[i]=a[i]=dl64(x+8*i);while(n>=128){FOR(i,16)w[i]=dl64(m+8*i);FOR(i,80){FOR(j,8)b[j]=a[j];t=a[7]+Sigma1
(a[4])+Ch(a[4],a[5],a[6])+K[i]+w[i%16];b[7]=t+Sigma0(a[0])+Maj(a[0],a[1],a[2]);b[3]+=t;FOR(j,8)a[(j+1)%8]=b[j];if(i%16==15)FOR(j,16)w[j]+=w[
(j+9)%16]+sigma0(w[(j+1)%16])+sigma1(w[(j+14)%16]);}FOR(i,8){a[i]+=z[i];z[i]=a[i];}m+=128;n-=128;}FOR(i,8)ts64(x+8*i,z[i]);return n;}static
const u8 iv[64]={0x6a,0x09,0xe6,0x67,0xf3,0xbc,0xc9,0x08,0xbb,0x67,0xae,0x85,0x84,0xca,0xa7,0x3b,0x3c,0x6e,0xf3,0x72,0xfe,0x94,0xf8,0x2b,
0xa5,0x4f,0xf5,0x3a,0x5f,0x1d,0x36,0xf1,0x51,0x0e,0x52,0x7f,0xad,0xe6,0x82,0xd1,0x9b,0x05,0x68,0x8c,0x2b,0x3e,0x6c,0x1f,0x1f,0x83,0xd9,0xab,
0xfb,0x41,0xbd,0x6b,0x5b,0xe0,0xcd,0x19,0x13,0x7e,0x21,0x79};int crypto_hash(u8*out,const u8*m,u64 n){u8 h[64],x[256];u64 i,b=n;FOR(i,64)h[i
]=iv[i];crypto_hashblocks(h,m,n);m+=n;n&=127;m-=n;FOR(i,256)x[i]=0;FOR(i,n)x[i]=m[i];x[n]=128;n=256-128*(n<112);x[n-9]=b>>61;ts64(x+n-8,b<<3
);crypto_hashblocks(h,x,n);FOR(i,64)out[i]=h[i];return 0;}sv add(gf p[4],gf q[4]){gf a,b,c,d,t,e,f,g,h;Z(a,p[1],p[0]);Z(t,q[1],q[0]);M(a,a,t
);A(b,p[0],p[1]);A(t,q[0],q[1]);M(b,b,t);M(c,p[3],q[3]);M(c,c,D2);M(d,p[2],q[2]);A(d,d,d);Z(e,b,a);Z(f,d,c);A(g,d,c);A(h,b,a);M(p[0],e,f);M(
p[1],h,g);M(p[2],g,f);M(p[3],e,h);}sv cswap(gf p[4],gf q[4],u8 b){int i;FOR(i,4)sel25519(p[i],q[i],b);}sv pack(u8*r,gf p[4]){gf tx,ty,zi;
inv25519(zi,p[2]);M(tx,p[0],zi);M(ty,p[1],zi);pack25519(r,ty);r[31]^=par25519(tx)<<7;}sv scalarmult(gf p[4],gf q[4],const u8*s){int i;
set25519(p[0],gf0);set25519(p[1],gf1);set25519(p[2],gf1);set25519(p[3],gf0);for(i=255;i>=0;--i){u8 b=(s[i/8]>>(i&7))&1;cswap(p,q,b);add(q,p)
;add(p,p);cswap(p,q,b);}}sv scalarbase(gf p[4],const u8*s){gf q[4];set25519(q[0],X);set25519(q[1],Y);set25519(q[2],gf1);M(q[3],X,Y);
scalarmult(p,q,s);}int crypto_sign_keypair(u8*pk,u8*sk){u8 d[64];gf p[4];int i;randombytes(sk,32);crypto_hash(d,sk,32);d[0]&=248;d[31]&=127;
d[31]|=64;scalarbase(p,d);pack(pk,p);FOR(i,32)sk[32+i]=pk[i];return 0;}static const u64 L[32]={0xed,0xd3,0xf5,0x5c,0x1a,0x63,0x12,0x58,0xd6,
0x9c,0xf7,0xa2,0xde,0xf9,0xde,0x14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0x10};sv modL(u8*r,i64 x[64]){i64 carry,i,j;for(i=63;i>=32;--i){carry=0;for
(j=i-32;j<i-12;++j){x[j]+=carry-16*x[i]*L[j-(i-32)];carry=(x[j]+128)>>8;x[j]-=carry<<8;}x[j]+=carry;x[i]=0;}carry=0;FOR(j,32){x[j]+=carry-(x
[31]>>4)*L[j];carry=x[j]>>8;x[j]&=255;}FOR(j,32)x[j]-=carry*L[j];FOR(i,32){x[i+1]+=x[i]>>8;r[i]=x[i]&255;}}sv reduce(u8*r){i64 x[64],i;FOR(i
,64)x[i]=(u64)r[i];FOR(i,64)r[i]=0;modL(r,x);}int crypto_sign(u8*sm,u64*smlen,const u8*m,u64 n,const u8*sk){u8 d[64],h[64],r[64];i64 i,j,x[
64];gf p[4];crypto_hash(d,sk,32);d[0]&=248;d[31]&=127;d[31]|=64;*smlen=n+64;FOR(i,n)sm[64+i]=m[i];FOR(i,32)sm[32+i]=d[32+i];crypto_hash(r,sm
+32,n+32);reduce(r);scalarbase(p,r);pack(sm,p);FOR(i,32)sm[i+32]=sk[i+32];crypto_hash(h,sm,n+64);reduce(h);FOR(i,64)x[i]=0;FOR(i,32)x[i]=(
u64)r[i];FOR(i,32)FOR(j,32)x[i+j]+=h[i]*(u64)d[j];modL(sm+32,x);return 0;}static int unpackneg(gf r[4],const u8 p[32]){gf t,chk,num,den,den2
,den4,den6;set25519(r[2],gf1);unpack25519(r[1],p);S(num,r[1]);M(den,num,D);Z(num,num,r[2]);A(den,r[2],den);S(den2,den);S(den4,den2);M(den6,
den4,den2);M(t,den6,num);M(t,t,den);pow2523(t,t);M(t,t,num);M(t,t,den);M(t,t,den);M(r[0],t,den);S(chk,r[0]);M(chk,chk,den);if(neq25519(chk,
num))M(r[0],r[0],I);S(chk,r[0]);M(chk,chk,den);if(neq25519(chk,num))return-1;if(par25519(r[0])==(p[31]>>7))Z(r[0],gf0,r[0]);M(r[3],r[0],r[1]
);return 0;}int crypto_sign_open(u8*m,u64*mlen,const u8*sm,u64 n,const u8*pk){int i;u8 t[32],h[64];gf p[4],q[4];*mlen= -1;if(n<64)return-1;
if(unpackneg(q,pk))return-1;FOR(i,n)m[i]=sm[i];FOR(i,32)m[i+32]=pk[i];crypto_hash(h,m,n);reduce(h);scalarmult(p,q,h);scalarbase(q,sm+32);add
(p,q);pack(t,p);n-=64;if(crypto_verify_32(sm,t)){FOR(i,n)m[i]=0;return-1;}FOR(i,n)m[i]=sm[i+64];*mlen=n;return 0;}
```

TweetNaCl online

`http://tweetnacl.cr.yp.to`

@tweetnacl