Review

Last time, we had some conversation about the change in how algebraic geometry, the areas of mathematics including the study of elliptic curves. Thirty years ago, algebraic geometry was considered among the most esoteric areas of mathematics. This began to change around 1970, when Goppa published of “A new class of linear error correcting codes,” Probl. Pered. Inform. Vol. 6 (1970), pp. 24 - 30. Goppa used abstract algebraic geometry to define a family of error correcting codes better than known codes. The paper is very difficult to read, and it took a while before the idea of using algebraic geometry in information theory caught on. But now it is everywhere. We have seen how elliptic curve cryptography is a rather straightforward generalization of Diffie–Hellman, in that both use repeated multiplication in a group; the only difference is the group chosen. More recent work has used a “Jacobian” of a “hyperelliptic curve” as the group. [One early version of the source code of the Unix kernel is reputed to have a block of code with the comment “You are not supposed to understand this;” this comment applies to the words in quotes.] These ideas have been studied deeply for many years, so there is a tremendous amount of mathematical machinery available to help in their analysis.

Richard mentioned the use of unicode spoofing in phishing attacks. Here is some more detail. As you probably know, Unicode is a system that “provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.” [from unicode.com] This is an opportunity for phishers, because there are characters in different languages that look like English characters; for example, the Cyrillic ‘p’, which has an ‘r’ sounds, is rendered just like Latin ‘p’. So, the unicode-encoded URL phishing.com (Cyrillic ‘p’) looks just like the unicode-encoded URL phishing.com, although in fact they are different. [I was surprised to see that the domain name phishing.com is available ... anyone want to camp there?]

We also talked about spoofing the “from” field in email. Richard fooled around with this and sent me an email that appeared to be from Bill Gates, although when I looked at the full header it was clearly identified as his. I tried to reply to it [insert sly grin emoticon here], but my reply bounced. this is a frequent phisher tactic. I am posting a phishing attack on the website. Notice that the true source of the attack is revealed in the expanded header. Also notice that the images come from the legitimate site, but the actual link is elsewhere.

Hash Security and Design Considerations

A hash function $H$ takes a variable length input and produces a fixed length output. This is generally done by processing the message $M$ in fixed length blocks; this generally requires
some padding, and the choice of padding is part of the algorithm. The output should look random.

It should be computationally infeasible to invert the function $H$, that is, it should be a one-way function. There are more subtle properties: since the input is so much larger than the output, it is likely that the function cannot be injective (or, as formerly called, one-to-one). Symbolically, it is likely that there are messages $M_1$ and $M_2$ such that $H(M_1) = H(M_2)$; this is called a collision, and the property of it being computationally difficult to find two such messages is called strong collision resistance. Collision resistance is the lynchpin of hash security, because the ability to generate a collision offers an opening to false authentication.

There is another kind of collision resistance: weak collision resistance means that given $y$ it is difficult to find a message $M$ with $y = H(M)$.

There are some subtleties to collisions, primarily based on the birthday paradox from probability. The usual phrasing of a birthday paradox is to take a room with 30 people or so and ask whether any two have the same birthday. This seems unlikely, because there are 366 possible birthdays, but that only addresses the probability that someone has your birthday. The “collision” probability in a group of 30 people is actually more than 0.5. (Actually, let’s ignore leap years in this.)

Why? The technique is to turn the question around and ask how likely is it that everyone has a different birthday. If the probability of no collision is $p$, then the probability of a collision is $1 - p$. Let $P(n, k)$ denote the probability of no collision with $n$ attributes and $k$ people.

So: pick a person. The probability that the next person has a different birthday is $364/365$, because every possible birthday but one is taken. The probability of no collision for the third person is $363/365$, because all but two dates are taken, so the probability of no collision among three people is $\frac{364 \cdot 363}{365^2}$. Generalizing, the probability of no collision among $k$ partygoers is $P(365, k) := 1 - \frac{365!}{(365 - k)!365^k}$. For example, $P(365, 23) > 0.5$, so the risk of collision is quite high. The point is that 23 is a lot smaller than 365.

When there are $2^{160}$ possible message digests, the number of messages needed to produce a collision is much smaller. This is called a birthday attack.

Stallings has a more detailed discussion of this and related phenomena in the appendix to Chapter 11. In general, the work required to generate a collision in an $n$-bit hash is approximately $2^n$.

---

**Message Digests**

We are going to study the SHA family, which is the current NIST standard. The NIST document is FIPS 180–2 (with change notice), which is available at the NIST website [http://csrc.nist.gov/cryptval/shs.htm](http://csrc.nist.gov/cryptval/shs.htm). Stallings removed the sections on MD4 and MD5 from the current edition of the text, because of security problems. MD4 and MD5
are still available on various Unix systems; the terminal fragments below show calls to MD5 and SHA1 for part of this lecture note file.

```
jimwolper$ md5 -q < LectureNotes/Lecture18.tex
df76cecd6b435202e4d649ae478fa6d0
jimwolper$ openssl dgst -sha1 < LectureNotes/Lecture18.tex
a4cb1b097ed9e084595b7c2a2ebcd9ebd5f3edd1
```

Many of these hash algorithms have a similar structure.

1. **pad**: The message length is seldom proper (usually an integer multiple of the digest size), so padding is added. Padding is added even when the message is the correct size. The padding may also include some representative of the length of the message.

2. **initialization**: The buffer has to begin with something. It could be zeroes, or ones, but it cannot be random or arbitrary (like, e.g., the buffer’s previous contents because the receiver needs to have the same initial buffer. SHA uses hexadecimal representations of prime numbers.

3. **Block-by-block processing**: this is the “round function.” SHA uses 80 rounds. The operations are a combination of XORs, shifts, permutations, or other logical operations. It is important that these operations be pretty close to the capabilities of most hardware (a couple of machine language instructions) for speed.

We will use the outline in Stallings to discuss this further.