

Secure Password Storage in SiteBuilder

Jonathan Oxeer <jon@ivt.com.au>

November 1st, 2007
Internet Vision Technologies
Melbourne, Victoria, AU

18cf7f57ff36142a4
73acdce6e602b03

Jonathan Oxer <jon@ivt.com.au>

November 1st, 2007
Internet Vision Technologies
Melbourne, Victoria, AU

“We want to make you aware that media of ours that contained a backup of a portion of the reddit database was stolen recently.

We wanted to alert you to the possibility that your username, password, and – in some cases – e-mail address may have been compromised.”

Steve Huffman, reddit.com

Lesson for site owners:

**Don't store
passwords in
plain text**

**Do we really need
to know user's
passwords?**

**No, we need to
know if *they*
know it!**

This is your password:

hammer

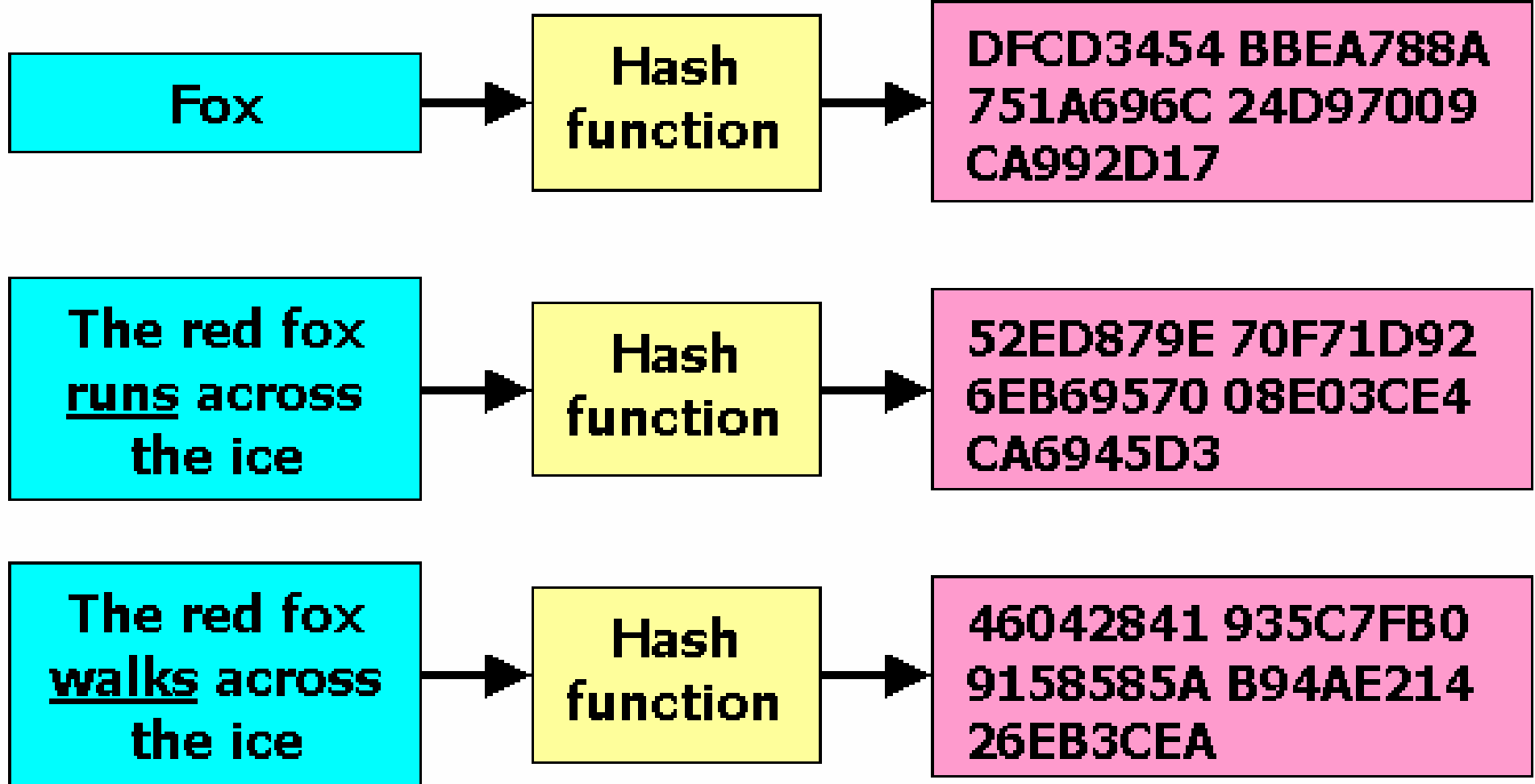
This is your password on hash:

**d58a27b9f79eb702
e1e514b0cdb4e254**

**A “hashing
algorithm” is
a one-way
calculation**

Input

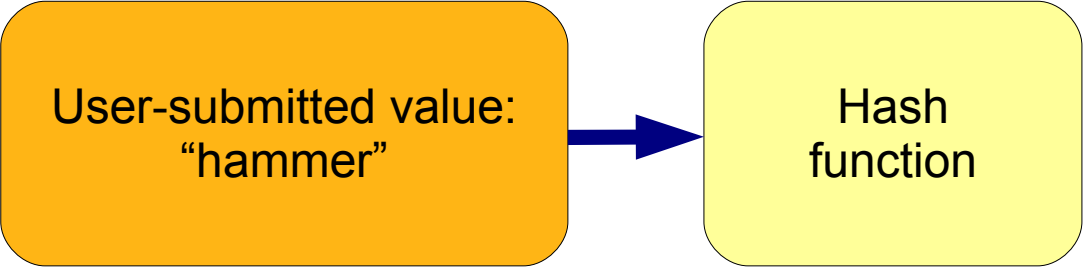
Hash sum

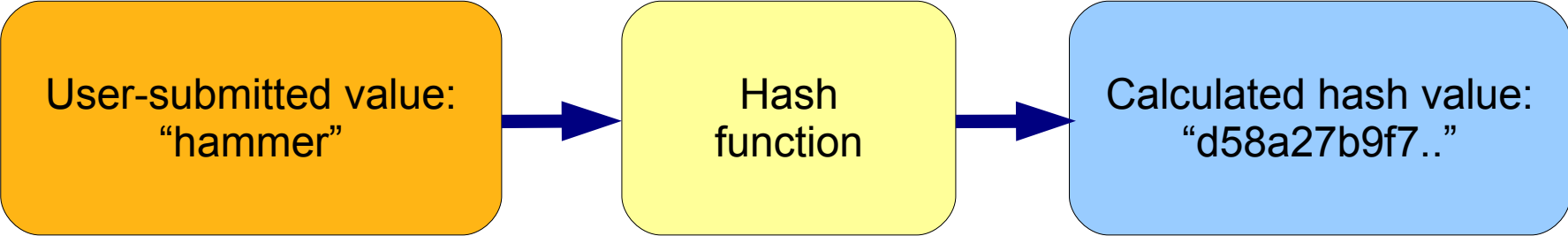


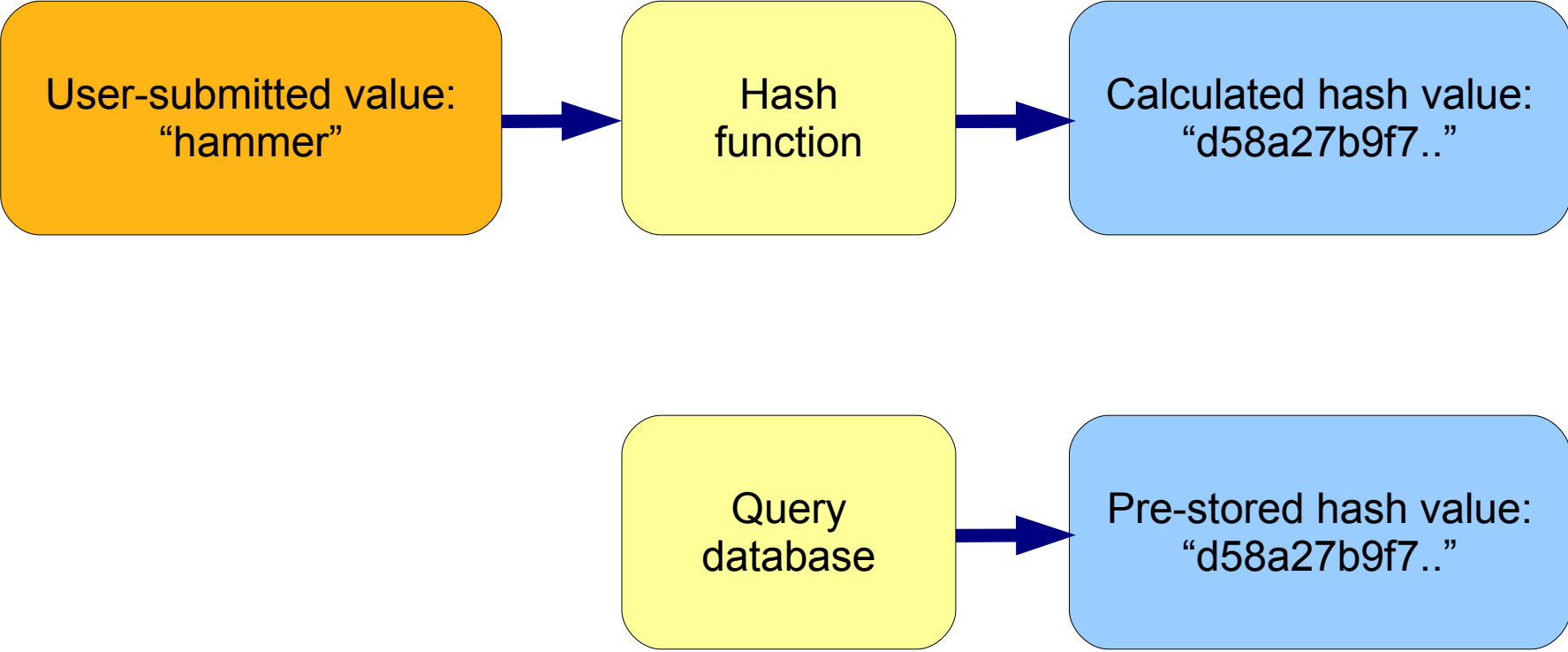
**Store the hashed
value, not the
plaintext**

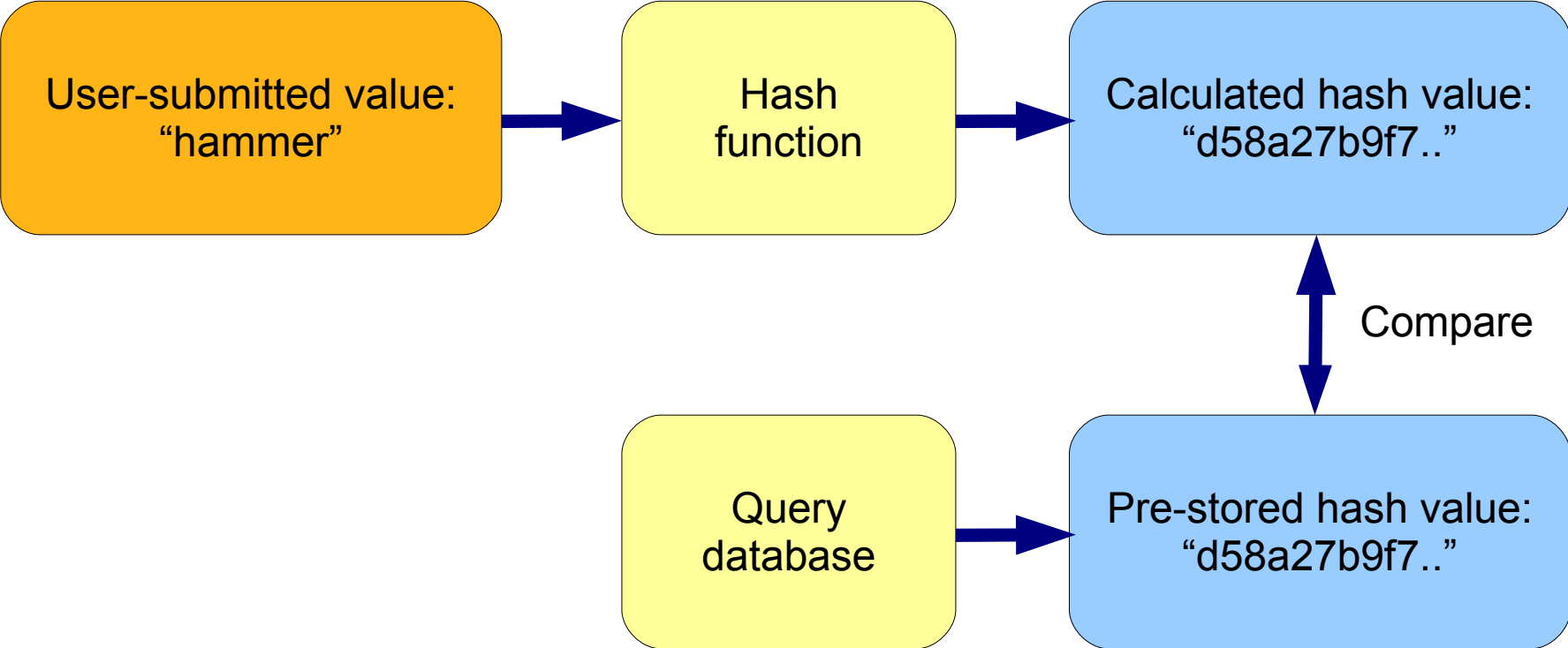
**On login: hash
the supplied
value and
compare hashes**

User-submitted value:
"hammer"









**Dictionary attack:
pre-compute hash
values for every
possible password**

echo "hammer" | md5sum
always equals

d58a27b9f79eb702
e1e514b0cdb4e254

So the input value for

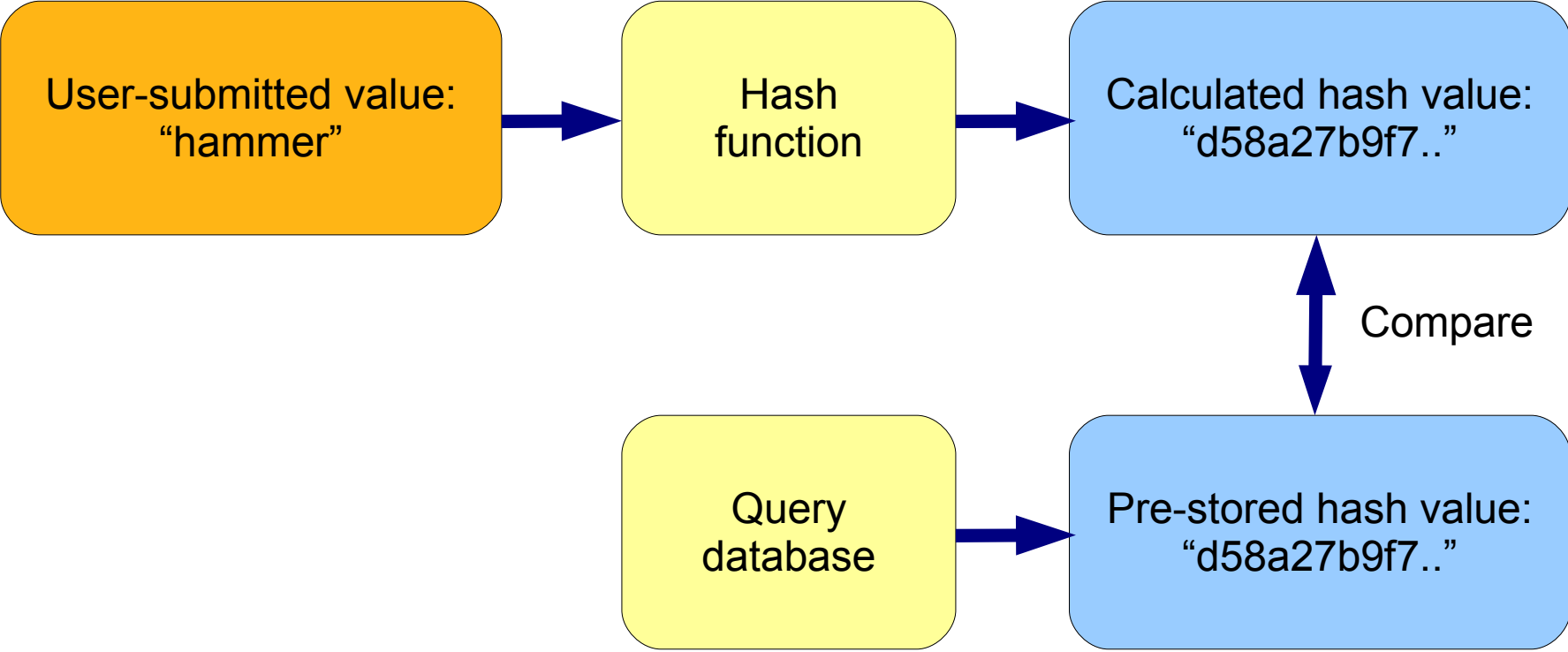
**d58a27b9f79eb702
e1e514b0cdb4e254**

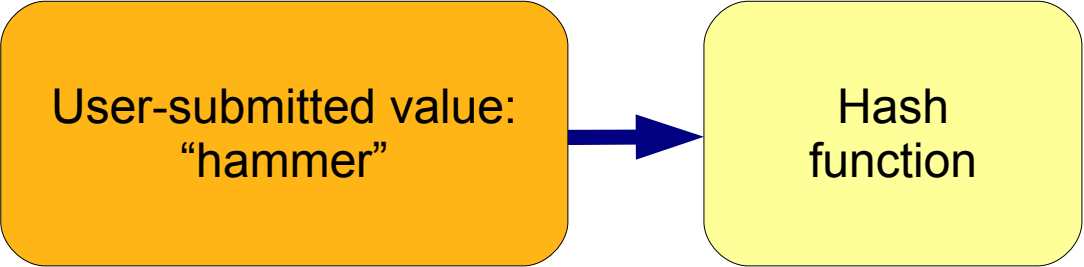
must have been 'hammer'

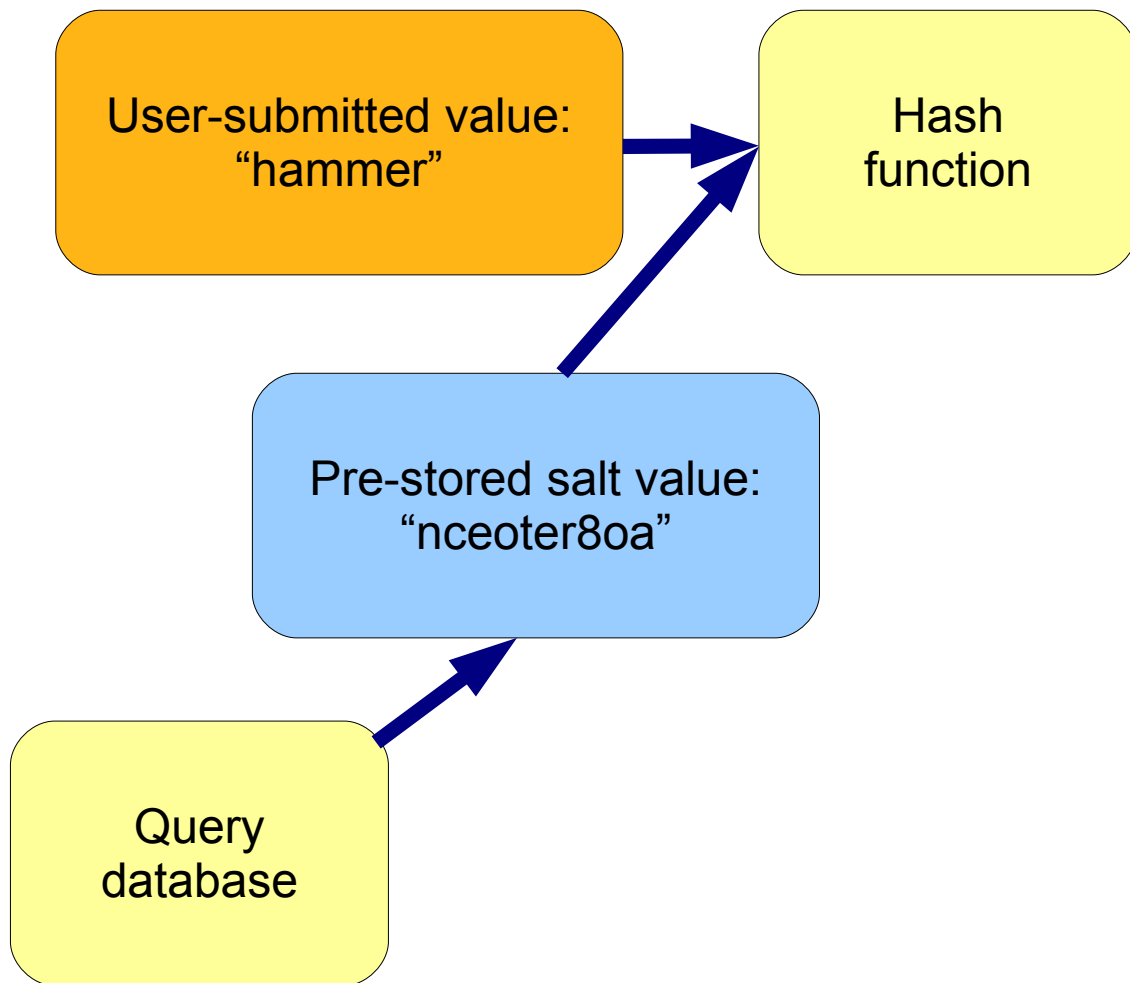
Dictionary attacks
pre-compute a
hash table for every
possible input value

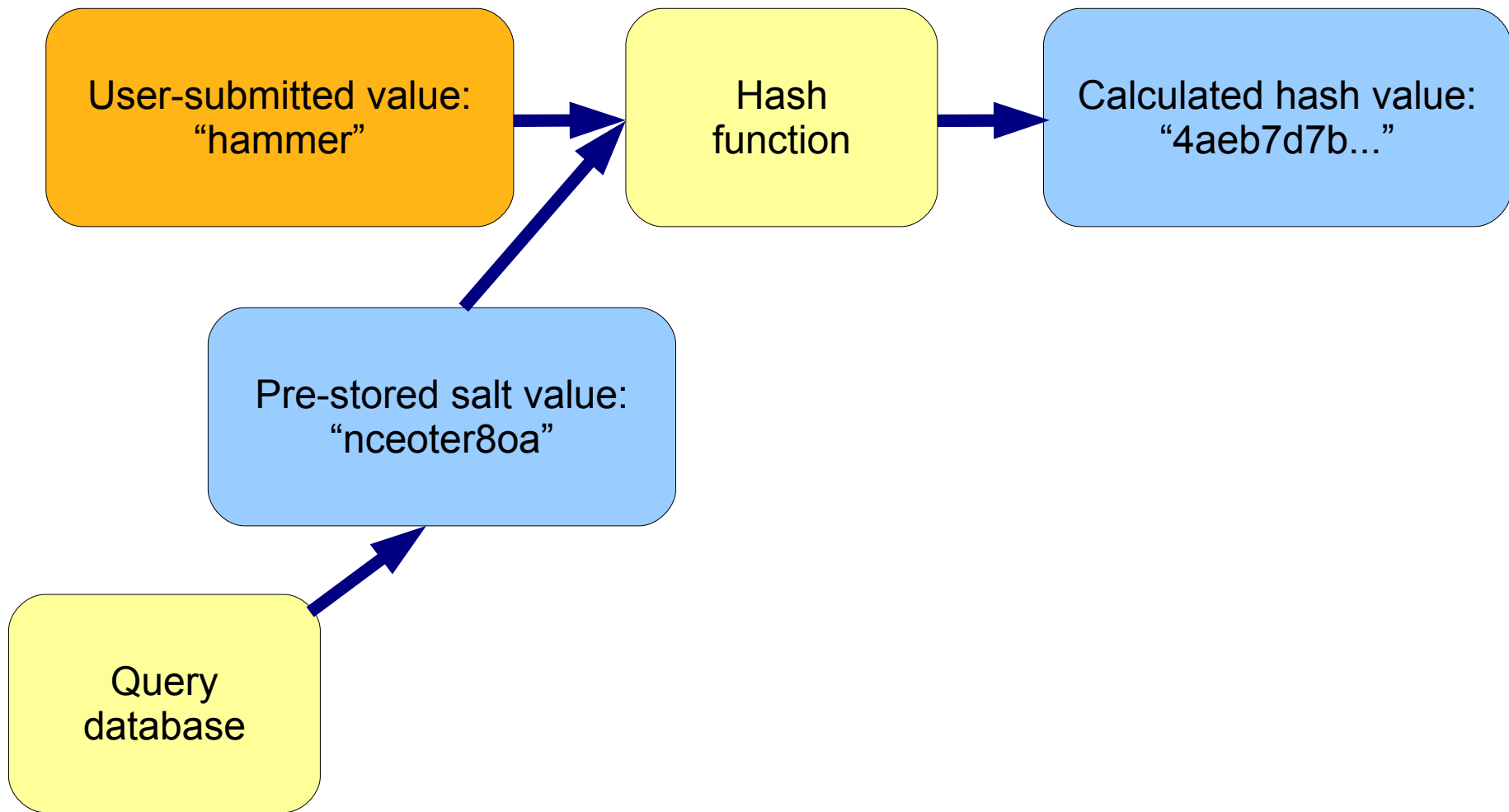
**Solution: “salt”
the plaintext
with a random
value first**

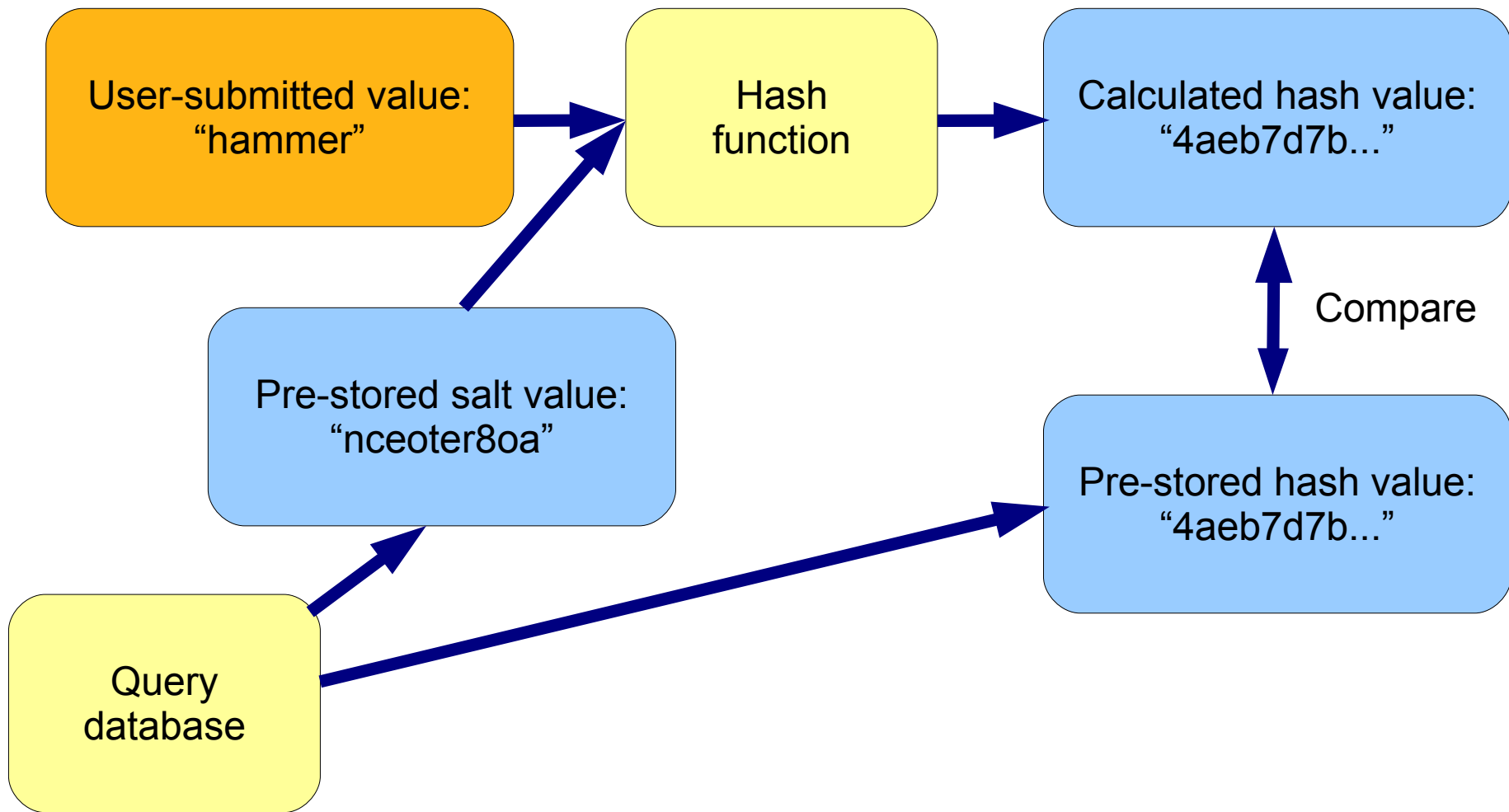
**Store the salt
value for later use
when validating
users**











**An attacker then
has to re-compute
their dictionary
for every attack**

**SiteBuilder uses
two fields:
'Password' and
*'PasswordSalt'***

**On a login request
SB checks for a
stored salt value**

**If salt found the
supplied value
is hashed
and compared
with password**

**If salt not found
plaintext value is
compared directly
with stored
password value**

**Then a salt is
generated, the
password is
hashed and both
values stored**

**Then a salt is
generated, the
password is
hashed and both
values stored**

**Next time the salt
will exist so pw
will be treated as
a hashed value**

**Progressive
encryption of
existing
passwords**

**Nice side effect:
update password
by simply writing
plaintext and
deleting salt**

```
UPDATE contacts  
SET `Password` = 'hammer',  
    `PasswordSalt` = "  
WHERE UserId = 123;
```

**On next login it
will be hashed
automatically**

Thankyou :-)

Questions? Comments? Insults?

Slides: jon.oxer.com.au/talks

Insults: `>/dev/null`

Questions: Jonathan Oxer jon@ivt.com.au